

MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.8

Tue Jan 26 2016 16:42:32

Contents

1	Introduction	1
1.1	MTK Concerns	1
1.2	MTK Wrappers	1
1.3	Contact, Support and Credits	2
1.3.1	Acknowledgements and Contributions	2
2	Referencing This Work	3
3	Read Me File and Installation Instructions	5
4	Programming Tools	9
5	Tests and Test Architectures	11
6	User Manual, References and Theory	13
7	Examples	15
8	Licensing and Modifications	17
9	Todo List	19
10	Bug List	21
11	Module Index	23
11.1	Modules	23
12	Namespace Index	25
12.1	Namespace List	25
13	Class Index	27
13.1	Class List	27
14	File Index	29

14.1 File List	29
15 Module Documentation	33
15.1 Roots.	33
15.1.1 Detailed Description	33
15.1.2 Typedef Documentation	34
15.1.2.1 Real	34
15.1.3 Variable Documentation	34
15.1.3.1 kCriticalOrderAccuracyDiv	34
15.1.3.2 kCriticalOrderAccuracyGrad	34
15.1.3.3 kDefaultMimeticThreshold	34
15.1.3.4 kDefaultOrderAccuracy	34
15.1.3.5 kDefaultTolerance	34
15.1.3.6 kOne	34
15.1.3.7 kTwo	35
15.1.3.8 kZero	35
15.2 Enumerations.	36
15.2.1 Detailed Description	36
15.2.2 Enumeration Type Documentation	36
15.2.2.1 DirInterp	36
15.2.2.2 FieldNature	36
15.2.2.3 MatrixOrdering	37
15.2.2.4 MatrixStorage	37
15.3 Execution tools.	38
15.3.1 Detailed Description	38
15.4 Data structures.	39
15.4.1 Detailed Description	39
15.5 Numerical methods.	40
15.5.1 Detailed Description	40
15.6 Grids.	41
15.6.1 Detailed Description	41
15.7 Mimetic operators.	42
15.7.1 Detailed Description	43
15.7.2 Typedef Documentation	43
15.7.2.1 CoefficientFunction0D	43
15.7.2.2 CoefficientFunction1D	43
15.7.2.3 CoefficientFunction2D	43

16 Namespace Documentation	45
16.1 mtk Namespace Reference	45
16.1.1 Function Documentation	48
16.1.1.1 operator<<	48
16.1.1.2 operator<<	48
16.1.1.3 operator<<	48
16.1.1.4 operator<<	48
16.1.1.5 operator<<	48
16.1.1.6 operator<<	48
16.1.1.7 operator<<	49
16.1.1.8 operator<<	49
16.1.1.9 saxpy_	49
16.1.1.10 sgels_	50
16.1.1.11 sgemm_	51
16.1.1.12 sgemv_	51
16.1.1.13 sgeqrf_	51
16.1.1.14 sgesv_	52
16.1.1.15 snrm2_	52
16.1.1.16 sormqr_	52
17 Class Documentation	55
17.1 mtk::BLASAdapter Class Reference	55
17.1.1 Detailed Description	56
17.1.2 Member Function Documentation	56
17.1.2.1 RealAXPY	56
17.1.2.2 RealDenseMM	57
17.1.2.3 RealDenseMV	58
17.1.2.4 RealDenseSM	60
17.1.2.5 RealNRM2	61
17.1.2.6 RelNorm2Error	62
17.2 mtk::Curl2D Class Reference	63
17.2.1 Detailed Description	65
17.2.2 Constructor & Destructor Documentation	65
17.2.2.1 Curl2D	65
17.2.2.2 Curl2D	65
17.2.2.3 ~Curl2D	65
17.2.3 Member Function Documentation	66

17.2.3.1	ConstructCurl2D	66
17.2.3.2	operator*	66
17.2.3.3	ReturnAsDenseMatrix	66
17.2.4	Member Data Documentation	67
17.2.4.1	curl_	67
17.2.4.2	mimetic_threshold_	67
17.2.4.3	order_accuracy_	67
17.3	mtk::DenseMatrix Class Reference	67
17.3.1	Detailed Description	70
17.3.2	Constructor & Destructor Documentation	70
17.3.2.1	DenseMatrix	70
17.3.2.2	DenseMatrix	70
17.3.2.3	DenseMatrix	71
17.3.2.4	DenseMatrix	72
17.3.2.5	DenseMatrix	72
17.3.2.6	~DenseMatrix	73
17.3.3	Member Function Documentation	73
17.3.3.1	data	73
17.3.3.2	GetValue	74
17.3.3.3	Kron	75
17.3.3.4	matrix_properties	76
17.3.3.5	num_cols	77
17.3.3.6	num_rows	78
17.3.3.7	operator=	79
17.3.3.8	operator==	80
17.3.3.9	OrderColMajor	81
17.3.3.10	OrderRowMajor	81
17.3.3.11	SetOrdering	82
17.3.3.12	SetValue	83
17.3.3.13	Transpose	84
17.3.3.14	WriteToFile	85
17.3.4	Friends And Related Function Documentation	85
17.3.4.1	operator<<	85
17.3.5	Member Data Documentation	85
17.3.5.1	data_	85
17.3.5.2	matrix_properties_	86
17.4	mtk::Div1D Class Reference	86

17.4.1 Detailed Description	89
17.4.2 Constructor & Destructor Documentation	89
17.4.2.1 Div1D	89
17.4.2.2 Div1D	89
17.4.2.3 ~Div1D	90
17.4.3 Member Function Documentation	90
17.4.3.1 AssembleOperator	90
17.4.3.2 coeffs_interior	90
17.4.3.3 ComputePreliminaryApproximations	90
17.4.3.4 ComputeRationalBasisNullSpace	91
17.4.3.5 ComputeStencilBoundaryGrid	92
17.4.3.6 ComputeStencilInteriorGrid	92
17.4.3.7 ComputeWeights	93
17.4.3.8 ConstructDiv1D	94
17.4.3.9 mim_bndy	94
17.4.3.10 num_bndy_coeffs	95
17.4.3.11 ReturnAsDenseMatrix	95
17.4.3.12 ReturnAsDimensionlessDenseMatrix	96
17.4.3.13 weights_cbs	96
17.4.3.14 weights_crs	96
17.4.4 Friends And Related Function Documentation	97
17.4.4.1 operator<<	97
17.4.5 Member Data Documentation	97
17.4.5.1 coeffs_interior_	97
17.4.5.2 dim_null_	97
17.4.5.3 divergence_	97
17.4.5.4 divergence_length_	97
17.4.5.5 mim_bndy_	97
17.4.5.6 mimetic_threshold_	97
17.4.5.7 minrow_	97
17.4.5.8 num_bndy_coeffs_	97
17.4.5.9 order_accuracy_	98
17.4.5.10 prem_apps_	98
17.4.5.11 rat_basis_null_space_	98
17.4.5.12 row_	98
17.4.5.13 sum_rows_mim_bndy_	98
17.4.5.14 weights_cbs_	98

17.4.5.15 weights_crs_	98
17.5 mtk::Div2D Class Reference	98
17.5.1 Detailed Description	100
17.5.2 Constructor & Destructor Documentation	100
17.5.2.1 Div2D	100
17.5.2.2 Div2D	100
17.5.2.3 ~Div2D	100
17.5.3 Member Function Documentation	101
17.5.3.1 ConstructDiv2D	101
17.5.3.2 ReturnAsDenseMatrix	101
17.5.4 Member Data Documentation	102
17.5.4.1 divergence_	102
17.5.4.2 mimetic_threshold_	102
17.5.4.3 order_accuracy_	102
17.6 mtk::Div3D Class Reference	102
17.6.1 Detailed Description	104
17.6.2 Constructor & Destructor Documentation	104
17.6.2.1 Div3D	104
17.6.2.2 Div3D	104
17.6.2.3 ~Div3D	104
17.6.3 Member Function Documentation	105
17.6.3.1 ConstructDiv3D	105
17.6.3.2 ReturnAsDenseMatrix	106
17.6.4 Member Data Documentation	106
17.6.4.1 divergence_	106
17.6.4.2 mimetic_threshold_	106
17.6.4.3 order_accuracy_	106
17.7 mtk::GLPKAdapter Class Reference	107
17.7.1 Detailed Description	107
17.7.2 Member Function Documentation	107
17.7.2.1 SolveSimplexAndCompare	108
17.8 mtk::Grad1D Class Reference	110
17.8.1 Detailed Description	113
17.8.2 Constructor & Destructor Documentation	113
17.8.2.1 Grad1D	113
17.8.2.2 Grad1D	113
17.8.2.3 ~Grad1D	114

17.8.3	Member Function Documentation	114
17.8.3.1	AssembleOperator	114
17.8.3.2	coeffs_interior	114
17.8.3.3	ComputePreliminaryApproximations	114
17.8.3.4	ComputeRationalBasisNullSpace	115
17.8.3.5	ComputeStencilBoundaryGrid	116
17.8.3.6	ComputeStencilInteriorGrid	116
17.8.3.7	ComputeWeights	117
17.8.3.8	ConstructGrad1D	117
17.8.3.9	mim_bndy	118
17.8.3.10	num_bndy_coeffs	119
17.8.3.11	ReturnAsDenseMatrix	119
17.8.3.12	ReturnAsDenseMatrix	120
17.8.3.13	ReturnAsDimensionlessDenseMatrix	121
17.8.3.14	weights_cbs	121
17.8.3.15	weights_crs	121
17.8.4	Friends And Related Function Documentation	121
17.8.4.1	operator<<	121
17.8.5	Member Data Documentation	122
17.8.5.1	coeffs_interior_	122
17.8.5.2	dim_null_	122
17.8.5.3	gradient_	122
17.8.5.4	gradient_length_	122
17.8.5.5	mim_bndy_	122
17.8.5.6	mimetic_threshold_	122
17.8.5.7	minrow_	122
17.8.5.8	num_bndy_approxs_	122
17.8.5.9	num_bndy_coeffs_	122
17.8.5.10	order_accuracy_	122
17.8.5.11	prem_apps_	122
17.8.5.12	rat_basis_null_space_	123
17.8.5.13	row_	123
17.8.5.14	weights_cbs_	123
17.8.5.15	weights_crs_	123
17.9	mtk::Grad2D Class Reference	123
17.9.1	Detailed Description	125
17.9.2	Constructor & Destructor Documentation	125

17.9.2.1	Grad2D	125
17.9.2.2	Grad2D	125
17.9.2.3	~Grad2D	125
17.9.3	Member Function Documentation	126
17.9.3.1	ConstructGrad2D	126
17.9.3.2	ReturnAsDenseMatrix	126
17.9.4	Member Data Documentation	127
17.9.4.1	gradient_	127
17.9.4.2	mimetic_threshold_	127
17.9.4.3	order_accuracy_	127
17.10	mtk::Grad3D Class Reference	127
17.10.1	Detailed Description	129
17.10.2	Constructor & Destructor Documentation	129
17.10.2.1	Grad3D	129
17.10.2.2	Grad3D	129
17.10.2.3	~Grad3D	129
17.10.3	Member Function Documentation	130
17.10.3.1	ConstructGrad3D	130
17.10.3.2	ReturnAsDenseMatrix	131
17.10.4	Member Data Documentation	131
17.10.4.1	gradient_	131
17.10.4.2	mimetic_threshold_	131
17.10.4.3	order_accuracy_	131
17.11	mtk::Interp1D Class Reference	132
17.11.1	Detailed Description	133
17.11.2	Constructor & Destructor Documentation	133
17.11.2.1	Interp1D	133
17.11.2.2	Interp1D	133
17.11.2.3	~Interp1D	133
17.11.3	Member Function Documentation	133
17.11.3.1	coeffs_interior	133
17.11.3.2	ConstructInterp1D	133
17.11.3.3	ReturnAsDenseMatrix	134
17.11.4	Friends And Related Function Documentation	134
17.11.4.1	operator<<	134
17.11.5	Member Data Documentation	134
17.11.5.1	coeffs_interior_	134

17.11.5.2	dir_interp_	135
17.11.5.3	order_accuracy_	135
17.12	mtk::Interp2D Class Reference	135
17.12.1	Detailed Description	137
17.12.2	Constructor & Destructor Documentation	137
17.12.2.1	Interp2D	137
17.12.2.2	Interp2D	137
17.12.2.3	~Interp2D	137
17.12.3	Member Function Documentation	137
17.12.3.1	ConstructInterp2D	137
17.12.3.2	ReturnAsDenseMatrix	138
17.12.4	Member Data Documentation	138
17.12.4.1	interpolator_	138
17.12.4.2	mimetic_threshold_	138
17.12.4.3	order_accuracy_	138
17.13	mtk::Lap1D Class Reference	138
17.13.1	Detailed Description	140
17.13.2	Constructor & Destructor Documentation	140
17.13.2.1	Lap1D	140
17.13.2.2	Lap1D	140
17.13.2.3	~Lap1D	140
17.13.3	Member Function Documentation	140
17.13.3.1	ConstructLap1D	140
17.13.3.2	data	142
17.13.3.3	delta	142
17.13.3.4	mimetic_threshold	142
17.13.3.5	order_accuracy	143
17.13.3.6	ReturnAsDenseMatrix	143
17.13.4	Friends And Related Function Documentation	144
17.13.4.1	operator<<	144
17.13.5	Member Data Documentation	144
17.13.5.1	delta_	144
17.13.5.2	laplacian_	144
17.13.5.3	laplacian_length_	144
17.13.5.4	mimetic_threshold_	145
17.13.5.5	order_accuracy_	145
17.14	mtk::Lap2D Class Reference	145

17.14.1 Detailed Description	147
17.14.2 Constructor & Destructor Documentation	147
17.14.2.1 Lap2D	147
17.14.2.2 Lap2D	147
17.14.2.3 ~Lap2D	147
17.14.3 Member Function Documentation	148
17.14.3.1 ConstructLap2D	148
17.14.3.2 data	148
17.14.3.3 ReturnAsDenseMatrix	149
17.14.4 Member Data Documentation	149
17.14.4.1 laplacian_	149
17.14.4.2 mimetic_threshold_	149
17.14.4.3 order_accuracy_	149
17.15mtk::Lap3D Class Reference	149
17.15.1 Detailed Description	151
17.15.2 Constructor & Destructor Documentation	151
17.15.2.1 Lap3D	151
17.15.2.2 Lap3D	151
17.15.2.3 ~Lap3D	151
17.15.3 Member Function Documentation	152
17.15.3.1 ConstructLap3D	152
17.15.3.2 data	152
17.15.3.3 operator*	153
17.15.3.4 ReturnAsDenseMatrix	153
17.15.4 Member Data Documentation	153
17.15.4.1 laplacian_	153
17.15.4.2 mimetic_threshold_	153
17.15.4.3 order_accuracy_	153
17.16mtk::LAPACKAdapter Class Reference	153
17.16.1 Detailed Description	154
17.16.2 Member Function Documentation	155
17.16.2.1 QRFactorDenseMatrix	155
17.16.2.2 SolveDenseSystem	156
17.16.2.3 SolveDenseSystem	157
17.16.2.4 SolveDenseSystem	158
17.16.2.5 SolveDenseSystem	159
17.16.2.6 SolveRectangularDenseSystem	160

17.17mtk::Matrix Class Reference	161
17.17.1 Detailed Description	164
17.17.2 Constructor & Destructor Documentation	164
17.17.2.1 Matrix	164
17.17.2.2 Matrix	165
17.17.2.3 ~Matrix	166
17.17.3 Member Function Documentation	166
17.17.3.1 abs_density	166
17.17.3.2 abs_sparsity	166
17.17.3.3 bandwidth	166
17.17.3.4 IncreaseNumNull	166
17.17.3.5 IncreaseNumZero	167
17.17.3.6 kl	167
17.17.3.7 ku	167
17.17.3.8 ld	167
17.17.3.9 num_cols	167
17.17.3.10num_non_null	168
17.17.3.11num_non_zero	168
17.17.3.12num_null	168
17.17.3.13num_rows	168
17.17.3.14num_values	169
17.17.3.15num_zero	169
17.17.3.16ordering	169
17.17.3.17rel_density	170
17.17.3.18rel_sparsity	170
17.17.3.19set_num_cols	170
17.17.3.20set_num_null	171
17.17.3.21set_num_rows	172
17.17.3.22set_num_zero	172
17.17.3.23set_ordering	173
17.17.3.24set_storage	174
17.17.3.25storage	174
17.17.4 Member Data Documentation	175
17.17.4.1 abs_density_	175
17.17.4.2 abs_sparsity_	175
17.17.4.3 bandwidth_	175
17.17.4.4 kl_	175

17.17.4.5 ku_	175
17.17.4.6 ld_	175
17.17.4.7 num_cols_	175
17.17.4.8 num_non_null_	176
17.17.4.9 num_non_zero_	176
17.17.4.10 num_null_	176
17.17.4.11 num_rows_	176
17.17.4.12 num_values_	176
17.17.4.13 num_zero_	176
17.17.4.14 ordering_	176
17.17.4.15 rel_density_	176
17.17.4.16 rel_sparsity_	176
17.17.4.17 storage_	176
17.18 mtk::Quad1D Class Reference	177
17.18.1 Detailed Description	178
17.18.2 Constructor & Destructor Documentation	178
17.18.2.1 Quad1D	178
17.18.2.2 Quad1D	178
17.18.2.3 ~Quad1D	179
17.18.3 Member Function Documentation	179
17.18.3.1 degree_approximation	179
17.18.3.2 Integrate	179
17.18.3.3 weights	179
17.18.4 Friends And Related Function Documentation	179
17.18.4.1 operator<<	179
17.18.5 Member Data Documentation	179
17.18.5.1 degree_approximation_	179
17.18.5.2 weights_	179
17.19 mtk::RobinBCDescriptor1D Class Reference	179
17.19.1 Detailed Description	181
17.19.2 Constructor & Destructor Documentation	182
17.19.2.1 RobinBCDescriptor1D	182
17.19.2.2 RobinBCDescriptor1D	182
17.19.2.3 ~RobinBCDescriptor1D	182
17.19.3 Member Function Documentation	182
17.19.3.1 highest_order_diff_east	182
17.19.3.2 highest_order_diff_west	182

17.19.3.3 ImposeOnGrid	183
17.19.3.4 ImposeOnLaplacianMatrix	183
17.19.3.5 PushBackEastCoeff	184
17.19.3.6 PushBackWestCoeff	185
17.19.3.7 set_east_condition	185
17.19.3.8 set_west_condition	186
17.19.4 Member Data Documentation	186
17.19.4.1 east_coefficients_	186
17.19.4.2 east_condition_	186
17.19.4.3 highest_order_diff_east_	186
17.19.4.4 highest_order_diff_west_	186
17.19.4.5 west_coefficients_	187
17.19.4.6 west_condition_	187
17.20mtk::RobinBCDescriptor2D Class Reference	187
17.20.1 Detailed Description	191
17.20.2 Constructor & Destructor Documentation	191
17.20.2.1 RobinBCDescriptor2D	191
17.20.2.2 RobinBCDescriptor2D	191
17.20.2.3 ~RobinBCDescriptor2D	191
17.20.3 Member Function Documentation	191
17.20.3.1 highest_order_diff_east	191
17.20.3.2 highest_order_diff_north	192
17.20.3.3 highest_order_diff_south	192
17.20.3.4 highest_order_diff_west	192
17.20.3.5 ImposeOnEastBoundaryNoSpace	192
17.20.3.6 ImposeOnEastBoundaryWithSpace	193
17.20.3.7 ImposeOnGrid	194
17.20.3.8 ImposeOnLaplacianMatrix	196
17.20.3.9 ImposeOnNorthBoundaryNoSpace	196
17.20.3.10 ImposeOnNorthBoundaryWithSpace	197
17.20.3.11 ImposeOnSouthBoundaryNoSpace	198
17.20.3.12 ImposeOnSouthBoundaryWithSpace	199
17.20.3.13 ImposeOnWestBoundaryNoSpace	200
17.20.3.14 ImposeOnWestBoundaryWithSpace	200
17.20.3.15 PushBackEastCoeff	201
17.20.3.16 PushBackNorthCoeff	201
17.20.3.17 PushBackSouthCoeff	202

17.20.3.18	PushBackWestCoeff	202
17.20.3.19	set_east_condition	203
17.20.3.20	set_north_condition	203
17.20.3.21	set_south_condition	204
17.20.3.22	set_west_condition	204
17.20.4	Member Data Documentation	205
17.20.4.1	east_coefficients_	205
17.20.4.2	east_condition_	205
17.20.4.3	highest_order_diff_east_	205
17.20.4.4	highest_order_diff_north_	205
17.20.4.5	highest_order_diff_south_	205
17.20.4.6	highest_order_diff_west_	205
17.20.4.7	north_coefficients_	205
17.20.4.8	north_condition_	205
17.20.4.9	south_coefficients_	206
17.20.4.10	south_condition_	206
17.20.4.11	west_coefficients_	206
17.20.4.12	west_condition_	206
17.21	mtk::RobinBCDescriptor3D Class Reference	206
17.21.1	Detailed Description	210
17.21.2	Constructor & Destructor Documentation	210
17.21.2.1	RobinBCDescriptor3D	210
17.21.2.2	RobinBCDescriptor3D	210
17.21.2.3	~RobinBCDescriptor3D	210
17.21.3	Member Function Documentation	210
17.21.3.1	highest_order_diff_west	210
17.21.3.2	ImposeOnEastBoundaryNoSpace	210
17.21.3.3	ImposeOnEastBoundaryWithSpace	211
17.21.3.4	ImposeOnGrid	211
17.21.3.5	ImposeOnLaplacianMatrix	211
17.21.3.6	ImposeOnNorthBoundaryNoSpace	211
17.21.3.7	ImposeOnNorthBoundaryWithSpace	211
17.21.3.8	ImposeOnSouthBoundaryNoSpace	212
17.21.3.9	ImposeOnSouthBoundaryWithSpace	212
17.21.3.10	ImposeOnWestBoundaryNoSpace	212
17.21.3.11	ImposeOnWestBoundaryWithSpace	212
17.21.3.12	PushBackWestCoeff	213

17.21.3.13set_west_condition	213
17.21.4 Member Data Documentation	213
17.21.4.1 bottom_coefficients_	213
17.21.4.2 bottom_condition_	213
17.21.4.3 east_coefficients_	213
17.21.4.4 east_condition_	213
17.21.4.5 highest_order_diff_bottom_	213
17.21.4.6 highest_order_diff_east_	213
17.21.4.7 highest_order_diff_north_	213
17.21.4.8 highest_order_diff_south_	214
17.21.4.9 highest_order_diff_top_	214
17.21.4.10highest_order_diff_west_	214
17.21.4.11north_coefficients_	214
17.21.4.12north_condition_	214
17.21.4.13south_coefficients_	214
17.21.4.14south_condition_	214
17.21.4.15top_coefficients_	214
17.21.4.16top_condition_	214
17.21.4.17west_coefficients_	214
17.21.4.18west_condition_	215
17.22mtk::Tools Class Reference	215
17.22.1 Detailed Description	216
17.22.2 Member Function Documentation	216
17.22.2.1 Assert	216
17.22.2.2 BeginUnitTestNo	216
17.22.2.3 EndUnitTestNo	216
17.22.2.4 Prevent	217
17.22.3 Member Data Documentation	217
17.22.3.1 begin_time_	217
17.22.3.2 duration_	217
17.22.3.3 test_number_	217
17.23mtk::UniStgGrid1D Class Reference	218
17.23.1 Detailed Description	220
17.23.2 Constructor & Destructor Documentation	220
17.23.2.1 UniStgGrid1D	220
17.23.2.2 UniStgGrid1D	220
17.23.2.3 UniStgGrid1D	220

17.23.2.4 ~UniStgGrid1D	221
17.23.3 Member Function Documentation	221
17.23.3.1 BindScalarField	221
17.23.3.2 BindVectorField	221
17.23.3.3 delta_x	222
17.23.3.4 discrete_domain_x	222
17.23.3.5 discrete_field	222
17.23.3.6 east_bndy_x	223
17.23.3.7 num_cells_x	223
17.23.3.8 west_bndy_x	224
17.23.3.9 WriteToFile	224
17.23.4 Friends And Related Function Documentation	224
17.23.4.1 operator<<	225
17.23.5 Member Data Documentation	225
17.23.5.1 delta_x_	225
17.23.5.2 discrete_domain_x_	225
17.23.5.3 discrete_field_	225
17.23.5.4 east_bndy_x_	225
17.23.5.5 nature_	225
17.23.5.6 num_cells_x_	225
17.23.5.7 west_bndy_x_	225
17.24mtk::UniStgGrid2D Class Reference	225
17.24.1 Detailed Description	228
17.24.2 Constructor & Destructor Documentation	229
17.24.2.1 UniStgGrid2D	229
17.24.2.2 UniStgGrid2D	229
17.24.2.3 UniStgGrid2D	229
17.24.2.4 ~UniStgGrid2D	229
17.24.3 Member Function Documentation	230
17.24.3.1 BindScalarField	230
17.24.3.2 BindVectorField	230
17.24.3.3 BindVectorFieldPComponent	231
17.24.3.4 BindVectorFieldQComponent	231
17.24.3.5 Bound	231
17.24.3.6 delta_x	232
17.24.3.7 delta_y	232
17.24.3.8 discrete_domain_x	233

17.24.3.9	discrete_domain_y	233
17.24.3.10	discrete_field	234
17.24.3.11	east_bndy	234
17.24.3.12	nature	235
17.24.3.13	north_bndy	236
17.24.3.14	num_cells_x	236
17.24.3.15	num_cells_y	237
17.24.3.16	Size	238
17.24.3.17	south_bndy	238
17.24.3.18	west_bndy	239
17.24.3.19	WriteToFile	240
17.24.4	Friends And Related Function Documentation	241
17.24.4.1	operator<<	241
17.24.5	Member Data Documentation	241
17.24.5.1	delta_x_	241
17.24.5.2	delta_y_	241
17.24.5.3	discrete_domain_x_	241
17.24.5.4	discrete_domain_y_	241
17.24.5.5	discrete_field_	242
17.24.5.6	east_bndy_	242
17.24.5.7	nature_	242
17.24.5.8	north_bndy_	242
17.24.5.9	num_cells_x_	242
17.24.5.10	num_cells_y_	242
17.24.5.11	south_bndy_	242
17.24.5.12	west_bndy_	242
17.25	mtk::UniStgGrid3D Class Reference	242
17.25.1	Detailed Description	246
17.25.2	Constructor & Destructor Documentation	246
17.25.2.1	UniStgGrid3D	246
17.25.2.2	UniStgGrid3D	246
17.25.2.3	UniStgGrid3D	246
17.25.2.4	~UniStgGrid3D	247
17.25.3	Member Function Documentation	247
17.25.3.1	BindScalarField	247
17.25.3.2	BindVectorField	248
17.25.3.3	BindVectorFieldPComponent	248

17.25.3.4 BindVectorFieldQComponent	249
17.25.3.5 BindVectorFieldRComponent	249
17.25.3.6 bottom_bndy	249
17.25.3.7 Bound	250
17.25.3.8 delta_x	250
17.25.3.9 delta_y	250
17.25.3.10 delta_z	250
17.25.3.11 discrete_domain_x	250
17.25.3.12 discrete_domain_y	250
17.25.3.13 discrete_domain_z	251
17.25.3.14 discrete_field	251
17.25.3.15 east_bndy	251
17.25.3.16 nature	251
17.25.3.17 north_bndy	251
17.25.3.18 num_cells_x	252
17.25.3.19 num_cells_y	252
17.25.3.20 num_cells_z	252
17.25.3.21 operator=	253
17.25.3.22 Size	253
17.25.3.23 south_bndy	253
17.25.3.24 top_bndy	253
17.25.3.25 west_bndy	254
17.25.3.26 WriteToFile	254
17.25.4 Friends And Related Function Documentation	255
17.25.4.1 operator<<	255
17.25.5 Member Data Documentation	255
17.25.5.1 bottom_bndy_	255
17.25.5.2 delta_x_	255
17.25.5.3 delta_y_	255
17.25.5.4 delta_z_	255
17.25.5.5 discrete_domain_x_	255
17.25.5.6 discrete_domain_y_	255
17.25.5.7 discrete_domain_z_	255
17.25.5.8 discrete_field_	256
17.25.5.9 east_bndy_	256
17.25.5.10 nature_	256
17.25.5.11 north_bndy_	256

17.25.5.12num_cells_x_	256
17.25.5.13num_cells_y_	256
17.25.5.14num_cells_z_	256
17.25.5.15south_bndy_	256
17.25.5.16top_bndy_	256
17.25.5.17west_bndy_	256
18 File Documentation	257
18.1 examples/curl_2d_angular_velocity/curl_2d_angular_velocity.cc File Reference	257
18.1.1 Detailed Description	257
18.1.2 Function Documentation	258
18.1.2.1 main	258
18.2 curl_2d_angular_velocity.cc	258
18.3 examples/diffusion_3d/diffusion_3d.cc File Reference	259
18.3.1 Detailed Description	260
18.3.2 Function Documentation	260
18.3.2.1 main	260
18.4 diffusion_3d.cc	260
18.5 examples/divergence_operators_1d/divergence_operators_1d.cc File Reference	262
18.5.1 Detailed Description	262
18.5.2 Function Documentation	262
18.5.2.1 main	262
18.6 divergence_operators_1d.cc	262
18.7 examples/divergence_operators_1d_mimetic_test/divergence_operators_1d_mimetic_test.cc File Reference	264
18.7.1 Detailed Description	264
18.7.2 Function Documentation	264
18.7.2.1 main	264
18.8 divergence_operators_1d_mimetic_test.cc	265
18.9 examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference	266
18.9.1 Detailed Description	266
18.9.2 Function Documentation	267
18.9.2.1 main	267
18.10minimalistic_poisson_1d.cc	267
18.11examples/poisson_1d/poisson_1d.cc File Reference	269
18.11.1 Detailed Description	269
18.11.2 Function Documentation	270
18.11.2.1 main	270

18.12	poisson_1d.cc	270
18.13	examples/poisson_2d/poisson_2d.cc File Reference	273
18.13.1	Detailed Description	273
18.13.2	Function Documentation	274
18.13.2.1	main	274
18.14	poisson_2d.cc	274
18.15	examples/pos_def_weights_1d/pos_def_weights_1d.cc File Reference	277
18.15.1	Detailed Description	277
18.15.2	Function Documentation	277
18.15.2.1	main	277
18.16	pos_def_weights_1d.cc	278
18.17	include/mtk.h File Reference	279
18.17.1	Detailed Description	280
18.18	mtk.h	281
18.19	include/mtk_blas_adapter.h File Reference	282
18.19.1	Detailed Description	283
18.20	mtk_blas_adapter.h	283
18.21	include/mtk_curl_2d.h File Reference	284
18.21.1	Detailed Description	286
18.22	mtk_curl_2d.h	286
18.23	include/mtk_dense_matrix.h File Reference	287
18.23.1	Detailed Description	288
18.24	mtk_dense_matrix.h	288
18.25	include/mtk_div_1d.h File Reference	290
18.25.1	Detailed Description	291
18.26	mtk_div_1d.h	291
18.27	include/mtk_div_2d.h File Reference	292
18.27.1	Detailed Description	294
18.28	mtk_div_2d.h	294
18.29	include/mtk_div_3d.h File Reference	295
18.29.1	Detailed Description	296
18.30	mtk_div_3d.h	296
18.31	include/mtk_enums.h File Reference	297
18.31.1	Detailed Description	298
18.32	mtk_enums.h	298
18.33	include/mtk_glpk_adapter.h File Reference	299
18.33.1	Detailed Description	300

18.34	mtk_glpk_adapter.h	301
18.35	include/mtk_grad_1d.h File Reference	302
18.35.1	Detailed Description	303
18.36	mtk_grad_1d.h	303
18.37	include/mtk_grad_2d.h File Reference	305
18.37.1	Detailed Description	306
18.38	mtk_grad_2d.h	306
18.39	include/mtk_grad_3d.h File Reference	307
18.39.1	Detailed Description	309
18.40	mtk_grad_3d.h	309
18.41	include/mtk_interp_1d.h File Reference	310
18.41.1	Detailed Description	311
18.42	mtk_interp_1d.h	311
18.43	include/mtk_interp_2d.h File Reference	313
18.43.1	Detailed Description	313
18.44	mtk_interp_2d.h	314
18.45	include/mtk_lap_1d.h File Reference	315
18.45.1	Detailed Description	316
18.46	mtk_lap_1d.h	316
18.47	include/mtk_lap_2d.h File Reference	318
18.47.1	Detailed Description	319
18.48	mtk_lap_2d.h	319
18.49	include/mtk_lap_3d.h File Reference	320
18.49.1	Detailed Description	322
18.50	mtk_lap_3d.h	322
18.51	include/mtk_lapack_adapter.h File Reference	323
18.51.1	Detailed Description	324
18.52	mtk_lapack_adapter.h	324
18.53	include/mtk_matrix.h File Reference	326
18.53.1	Detailed Description	326
18.54	mtk_matrix.h	327
18.55	include/mtk_quad_1d.h File Reference	328
18.55.1	Detailed Description	329
18.56	mtk_quad_1d.h	330
18.57	include/mtk_robin_bc_descriptor_1d.h File Reference	331
18.57.1	Detailed Description	332
18.58	mtk_robin_bc_descriptor_1d.h	333

18.59	include/mtk_robin_bc_descriptor_2d.h File Reference	334
18.59.1	Detailed Description	336
18.60	mtk_robin_bc_descriptor_2d.h	336
18.61	include/mtk_robin_bc_descriptor_3d.h File Reference	338
18.61.1	Detailed Description	340
18.62	mtk_robin_bc_descriptor_3d.h	340
18.63	include/mtk_roots.h File Reference	342
18.63.1	Detailed Description	343
18.64	mtk_roots.h	344
18.65	include/mtk_tools.h File Reference	345
18.65.1	Detailed Description	345
18.66	mtk_tools.h	346
18.67	include/mtk_uni_stg_grid_1d.h File Reference	347
18.67.1	Detailed Description	348
18.68	mtk_uni_stg_grid_1d.h	348
18.69	include/mtk_uni_stg_grid_2d.h File Reference	349
18.69.1	Detailed Description	350
18.70	mtk_uni_stg_grid_2d.h	351
18.71	include/mtk_uni_stg_grid_3d.h File Reference	352
18.71.1	Detailed Description	354
18.72	mtk_uni_stg_grid_3d.h	354
18.73	Makefile.inc File Reference	356
18.74	Makefile.inc	356
18.75	README.md File Reference	359
18.76	README.md	359
18.77	src/mtk_blas_adapter.cc File Reference	361
18.77.1	Detailed Description	362
18.78	mtk_blas_adapter.cc	363
18.79	src/mtk_curl_2d.cc File Reference	367
18.79.1	Detailed Description	367
18.80	mtk_curl_2d.cc	368
18.81	src/mtk_dense_matrix.cc File Reference	370
18.82	mtk_dense_matrix.cc	370
18.83	src/mtk_div_1d.cc File Reference	377
18.83.1	Detailed Description	378
18.84	mtk_div_1d.cc	379
18.85	src/mtk_div_2d.cc File Reference	397

18.85.1 Detailed Description	397
18.86mtk_div_2d.cc	398
18.87src/mtk_div_3d.cc File Reference	399
18.87.1 Detailed Description	400
18.88mtk_div_3d.cc	400
18.89src/mtk_glpk_adapter.cc File Reference	403
18.89.1 Detailed Description	403
18.90mtk_glpk_adapter.cc	403
18.91src/mtk_grad_1d.cc File Reference	407
18.91.1 Detailed Description	408
18.92mtk_grad_1d.cc	409
18.93src/mtk_grad_2d.cc File Reference	428
18.93.1 Detailed Description	428
18.94mtk_grad_2d.cc	429
18.95src/mtk_grad_3d.cc File Reference	430
18.95.1 Detailed Description	431
18.96mtk_grad_3d.cc	431
18.97src/mtk_interp_1d.cc File Reference	433
18.97.1 Detailed Description	434
18.98mtk_interp_1d.cc	434
18.99src/mtk_lap_1d.cc File Reference	437
18.99.1 Detailed Description	437
18.100mtk_lap_1d.cc	437
18.100src/mtk_lap_2d.cc File Reference	442
18.101.1 Detailed Description	442
18.102mtk_lap_2d.cc	442
18.103src/mtk_lap_3d.cc File Reference	444
18.103.1 Detailed Description	444
18.104mtk_lap_3d.cc	445
18.105src/mtk_lapack_adapter.cc File Reference	446
18.105.1 Detailed Description	447
18.106mtk_lapack_adapter.cc	448
18.107src/mtk_matrix.cc File Reference	455
18.107.1 Detailed Description	456
18.108mtk_matrix.cc	456
18.109src/mtk_robin_bc_descriptor_1d.cc File Reference	460
18.109.1 Detailed Description	460

18.110	mtk_robin_bc_descriptor_1d.cc	461
18.111	src/mtk_robin_bc_descriptor_2d.cc File Reference	463
18.111.1	Detailed Description	464
18.112	mtk_robin_bc_descriptor_2d.cc	465
18.113	src/mtk_tools.cc File Reference	474
18.113.1	Detailed Description	474
18.114	mtk_tools.cc	474
18.115	src/mtk_uni_stg_grid_1d.cc File Reference	476
18.115.1	Detailed Description	476
18.116	mtk_uni_stg_grid_1d.cc	477
18.117	src/mtk_uni_stg_grid_2d.cc File Reference	480
18.117.1	Detailed Description	481
18.118	mtk_uni_stg_grid_2d.cc	481
18.119	src/mtk_uni_stg_grid_3d.cc File Reference	487
18.119.1	Detailed Description	488
18.120	mtk_uni_stg_grid_3d.cc	488
18.121	tests/mtk_blas_adapter_test.cc File Reference	494
18.121.1	Detailed Description	494
18.121.2	Function Documentation	494
18.121.2.1	main	494
18.122	mtk_blas_adapter_test.cc	495
18.123	tests/mtk_dense_matrix_test.cc File Reference	496
18.123.1	Detailed Description	496
18.123.2	Function Documentation	497
18.123.2.1	main	497
18.124	mtk_dense_matrix_test.cc	497
18.125	tests/mtk_div_1d_test.cc File Reference	501
18.125.1	Detailed Description	502
18.125.2	Function Documentation	502
18.125.2.1	main	502
18.126	mtk_div_1d_test.cc	502
18.127	tests/mtk_div_2d_test.cc File Reference	506
18.127.1	Detailed Description	506
18.127.2	Function Documentation	507
18.127.2.1	main	507
18.128	mtk_div_2d_test.cc	507
18.129	tests/mtk_div_3d_test.cc File Reference	508

18.129.1	Detailed Description	509
18.129.2	Function Documentation	509
18.129.2.1	main	509
18.130	mtk_div_3d_test.cc	509
18.131	tests/mtk_glpk_adapter_test.cc File Reference	511
18.131.1	Detailed Description	512
18.131.2	Function Documentation	512
18.131.2.1	main	512
18.132	mtk_glpk_adapter_test.cc	512
18.133	tests/mtk_grad_1d_test.cc File Reference	513
18.133.1	Detailed Description	513
18.133.2	Function Documentation	514
18.133.2.1	main	514
18.134	mtk_grad_1d_test.cc	514
18.135	tests/mtk_grad_2d_test.cc File Reference	518
18.135.1	Detailed Description	518
18.135.2	Function Documentation	518
18.135.2.1	main	519
18.136	mtk_grad_2d_test.cc	519
18.137	tests/mtk_grad_3d_test.cc File Reference	520
18.137.1	Detailed Description	521
18.137.2	Function Documentation	521
18.137.2.1	main	521
18.138	mtk_grad_3d_test.cc	521
18.139	tests/mtk_interp_1d_test.cc File Reference	523
18.139.1	Detailed Description	524
18.139.2	Function Documentation	524
18.139.2.1	main	524
18.140	mtk_interp_1d_test.cc	524
18.141	tests/mtk_lap_1d_test.cc File Reference	525
18.141.1	Detailed Description	526
18.141.2	Function Documentation	526
18.141.2.1	main	526
18.142	mtk_lap_1d_test.cc	526
18.143	tests/mtk_lap_2d_test.cc File Reference	529
18.143.1	Detailed Description	529
18.143.2	Function Documentation	529

18.143.2.1main	529
18.144ntk_lap_2d_test.cc	530
18.145tests/mtk_lap_3d_test.cc File Reference	531
18.145.1Detailed Description	532
18.145.2Function Documentation	532
18.145.2.1main	532
18.146ntk_lap_3d_test.cc	532
18.147tests/mtk_lapack_adapter_test.cc File Reference	534
18.147.1Detailed Description	535
18.147.2Function Documentation	535
18.147.2.1main	535
18.148ntk_lapack_adapter_test.cc	535
18.149tests/mtk_robin_bc_descriptor_2d_test.cc File Reference	536
18.149.1Detailed Description	537
18.149.2Function Documentation	537
18.149.2.1main	537
18.150ntk_robin_bc_descriptor_2d_test.cc	537
18.151tests/mtk_uni_stg_grid_1d_test.cc File Reference	539
18.151.1Detailed Description	540
18.151.2Function Documentation	540
18.151.2.1main	540
18.152ntk_uni_stg_grid_1d_test.cc	540
18.153tests/mtk_uni_stg_grid_2d_test.cc File Reference	543
18.153.1Detailed Description	543
18.153.2Function Documentation	543
18.153.2.1main	543
18.154ntk_uni_stg_grid_2d_test.cc	543
18.155tests/mtk_uni_stg_grid_3d_test.cc File Reference	546
18.155.1Detailed Description	546
18.155.2Function Documentation	547
18.155.2.1main	547
18.156ntk_uni_stg_grid_3d_test.cc	547

Chapter 1

Introduction

We define numerical methods that are based on discretizations preserving the properties of their continuous counterparts to be **mimetic**.

The **Mimetic Methods Toolkit (MTK)** is a C++11 library for mimetic numerical methods. It is a set of classes for **mimetic interpolation**, **mimetic quadratures**, and **mimetic finite difference** methods for the **numerical solution of ordinary and partial differential equations**.

1.1 MTK Concerns

Since collaborative development efforts are definitely important in achieving the level of generality we intend the library to possess, we have divided the library's source code according to the designated purpose the classes possess within the library. These divisions (or **concerns**) are grouped by layers, and are hierarchically related by the dependence they have among them.

One concern is said to depend on another one, if the classes the first concern includes, rely on the classes the second concern includes.

In order of dependence these are:

1. Roots.
2. Enumerations.
3. Tools.
4. Data Structures.
5. Numerical Methods.
6. Grids.
7. Mimetic Operators.

1.2 MTK Wrappers

The MTK collection of wrappers is:

1. MMTK: MATLAB wrappers collection for MTK; intended for sequential computations.

Others are being strongly considered.

1.3 Contact, Support and Credits

The GitHub repository is: <https://github.com/ejspeiro/MTK>

The MTK is developed by researchers and adjuncts to the [Computational Science Research Center \(CSRC\)](#) at [San Diego State University \(SDSU\)](#).

Currently the developers are:

- **Eduardo J. Sanchez, PhD** - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu) - ejspeiro
- Jose E. Castillo, PhD - [jcastillo at mail dot sdsu dot edu](mailto:jcastillo@mail.sdsu.edu)
- Guillermo F. Miranda, PhD - [unigrav at hotmail dot com](mailto:unigrav@hotmail.com)
- Christopher P. Paolini, PhD - [paolini at engineering dot sdsu dot edu](mailto:paolini@engineering.sdsu.edu)
- Angel Boada.
- Johnny Corbino.
- Raul Vargas-Navarro.

1.3.1 Acknowledgements and Contributions

The authors would like to acknowledge valuable advising, contributions and feedback, from research personnel at the Computational Science Research Center at San Diego State University, which were vital to the fruition of this work. Specifically, our thanks go to (alphabetical order):

1. Mohammad Abouali, Ph.D.
2. Dany De Cecchis, Ph.D.
3. Otilio Rojas, Ph.D.
4. Julia Rossi.

Chapter 2

Referencing This Work

Please reference this work as follows:

```
@article{Sanchez2014308,
  title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
Finite Differences ",
  journal = "Journal of Computational and Applied Mathematics ",
  volume = "270",
  number = "",
  pages = "308 - 322",
  year = "2014",
  note = "Fourth International Conference on Finite Element Methods in
Engineering and Sciences (FEMTEC 2013) ",
  issn = "0377-0427",
  doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
  url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
  author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
  keywords = "Object-oriented development",
  keywords = "Partial differential equations",
  keywords = "Application programming interfaces",
  keywords = "Mimetic Finite Differences "
}

@Inbook{Sanchez2015,
  author="Sanchez, Eduardo and Paolini, Christopher and Blomgren, Peter
and Castillo, Jose",
  editor="Kirby, M. Robert and Berzins, Martin and Hesthaven, S. Jan",
  chapter="Algorithms for Higher-Order Mimetic Operators",
  title="Spectral and High Order Methods for Partial Differential Equations
ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014,
Salt Lake City, Utah, USA",
  year="2015",
  publisher="Springer International Publishing",
  address="Cham",
  pages="425--434",
  isbn="978-3-319-19800-2",
  doi="10.1007/978-3-319-19800-2_39",
  url="http://dx.doi.org/10.1007/978-3-319-19800-2_39"
}
```


Chapter 3

Read Me File and Installation Instructions

The Mimetic Methods Toolkit (MTK)

By: **Eduardo J. Sanchez**, PhD - esanchez at mail dot sdsu dot edu

1. Description

We define numerical methods that are based on discretizations preserving the properties of their continuous counterparts to be **mimetic**.

The **Mimetic Methods Toolkit (MTK)** is a C++11 library for mimetic numerical methods. It is a set of classes for **mimetic interpolation**, **mimetic quadratures**, and **mimetic finite difference** methods for the **numerical solution of ordinary and partial differential equations**.

2. Dependencies

This README file assumes all of these dependencies are installed in the following folder:

```
```\n$(HOME)/Libraries/\n```
```

In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK routines for the internal computation on some of the layers. However, ATLAS requires both BLAS and LAPACK in order to create their optimized distributions. Therefore, the following dependencies tree arises:

#### ### For Linux:

1. LAPACK - Available from: <http://www.netlib.org/lapack/>
  1. BLAS - Available from: <http://www.netlib.org/blas/>
2. GLPK - Available from: <https://www.gnu.org/software/glpk/>
3. (Optional) ATLAS - Available from: <http://math-atlas.sourceforge.net/>
  1. LAPACK - Available from: <http://www.netlib.org/lapack/>
  1. BLAS - Available from: [http://www.netlib.org/blas](http://www.netlib.org/blas/)
4. (Optional) Valgrind - Available from: <http://valgrind.org/>
5. (Optional) Doxygen - Available from <http://www.stack.nl/~dimitri/doxygen/>

#### ### For OS X:

1. GLPK - Available from: <https://www.gnu.org/software/glpk/>

### ## 3. Installation

### ### PART 1. CONFIGURATION OF THE MAKEFILE.

The following steps are required to build and test the MTK. Please use the accompanying 'Makefile.inc' file, which should provide a solid template to start with. The following command provides help on the options for make:

```
'''
$ make help

Makefile for the MTK.

Options are:
- all: builds the library, the tests, and examples.
- mtklib: builds the library.
- test: builds the test files.
- example: builds the examples.

- testall: runs all the tests.

- gendoc: generates the documentation for the library.

- clean: cleans all the generated files.
- cleanlib: cleans the generated archive and object files.
- cleantest: cleans the generated tests executables.
- cleanexample: cleans the generated examples executables.

'''
```

### ### PART 2. BUILD THE LIBRARY.

```
'''
$ make
'''

If successful you'll read (before building the tests and examples):
'''
----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
'''
```

### ## 4. Contact, Support, and Credits

The GitHub repository is: <https://github.com/ejspeiro/MTK>

The MTK is developed by researchers and adjuncts to the [Computational Science Research Center (CSRC)] (<http://www.csrc.sdsu.edu/>) at [San Diego State University (SDSU)] (<http://www.sdsu.edu/>).

Currently the developers are:

- \*\*Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu\*\* - @ejspeiro
- Jose E. Castillo, PhD - jcastillo at mail dot sdsu dot edu
- Guillermo F. Miranda, PhD - unigrav at hotmail dot com
- Christopher P. Paolini, PhD - paolini at engineering dot sdsu dot edu
- Angel Boada.
- Johnny Corbino.
- Raul Vargas-Navarro.

#### ### 4.1. Acknowledgements and Contributions

The authors would like to acknowledge valuable advising, feedback, and actual contributions from research personnel at the Computational Science Research Center (CSRC) at San Diego State University (SDSU). Their input was important to the fruition of this work. Specifically, our thanks go to (alphabetical order):

- # Mohammad Abouali, PhD
- # Dany De Cecchis, PhD

---

```
-# Otilio Rojas, PhD
-# Julia Rossi.
```

## ## 5. Referencing This Work

Please reference this work as follows:

Please reference this work as follows:

```
```
```

```
@article{Sanchez2014308,
  title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
Finite Differences ",
  journal = "Journal of Computational and Applied Mathematics ",
  volume = "270",
  number = "",
  pages = "308 - 322",
  year = "2014",
  note = "Fourth International Conference on Finite Element Methods in
Engineering and Sciences (FEMTEC 2013) ",
  issn = "0377-0427",
  doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
  url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
  author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
  keywords = "Object-oriented development",
  keywords = "Partial differential equations",
  keywords = "Application programming interfaces",
  keywords = "Mimetic Finite Differences "
}
```

```
@Inbook{Sanchez2015,
  author="Sanchez, Eduardo and Paolini, Christopher and Blomgren, Peter
and Castillo, Jose",
  editor="Kirby, M. Robert and Berzins, Martin and Hesthaven, S. Jan",
  chapter="Algorithms for Higher-Order Mimetic Operators",
  title="Spectral and High Order Methods for Partial Differential Equations
ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014,
Salt Lake City, Utah, USA",
  year="2015",
  publisher="Springer International Publishing",
  address="Cham",
  pages="425--434",
  isbn="978-3-319-19800-2",
  doi="10.1007/978-3-319-19800-2_39",
  url="http://dx.doi.org/10.1007/978-3-319-19800-2_39"
}
```
```

Finally, please feel free to contact me with suggestions or corrections:

**\*\*Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu\*\* - @ejspeiro**

Thanks and happy coding!



## Chapter 4

# Programming Tools

The development of MTK has been made possible through the use of the following applications:

1. Editor: Kate - KDE Advanced Text Editor. Version 3.13.3. Using KDE Development Platform 4.13.3 (C) 2000-2005 The Kate Authors.
2. Debugger: GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1. Copyright (C) 2014 Free Software Foundation, Inc.
3. Memory Profiler: valgrind-3.10.0.SVN.

See the section on test architectures for information about operating systems and compilers used.



## Chapter 5

# Tests and Test Architectures

Tests are given in the `files list` section. They are provided in the `/tests/` folder within the distributed software.

In this page we intend to make a summary of all of the architectures in where the MTK has been tested. The MTK is intended to be as portable as possible throughout architectures. The following architectures have provided flawless installations of the API and correct execution of the tests and the examples:

1. Intel(R) Pentium(R) M CPU 1.73 GHz 2048 KB of cache and stepping of 8.  
Linux 3.2.0-23-generic-pae #36-Ubuntu SMP i386 GNU/Linux  
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
2. Intel(R) Core(TM) i7-4700MQ CPU 2.40 GHz 6144 KB of cache and stepping of 3.  
Linux 3.13.0-67-generic #110-Ubuntu SMP x86\_64 GNU/Linux  
gcc version 4.8.4 (Ubuntu 4.4.4-2ubuntu1~14.04)
3. Intel(R) Core(TM) i7-4600U CPU 2.10 GHz 4096 KB of cache and a stepping of 1.  
Linux 3.16.7-29-desktop #1 SMP PREEMPT (6be6a97) x86\_64 GNU/Linux  
openSUSE 13.2 (Harlequin) (x86\_64)  
gcc (SUSE Linux) 4.8.3 20140627 [gcc-4\_8-branch revision 212064]

Further architectures will be tested!





## Chapter 6

# User Manual, References and Theory

The main source of references for this work can be found in:

<http://www.csrc.sdsu.edu/mimetic-book/>

However, a .PDF copy of this manual can be found [here](#).



## Chapter 7

# Examples

Examples are given in the `files list` section. They are provided in the `/examples/` folder within the distributed software.



## Chapter 8

# Licensing and Modifications

Copyright (C) 2015, Computational Science Research Center, San Diego State University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Modifications to source code should be reported to: [esanchez@mail.sdsu.edu](mailto:esanchez@mail.sdsu.edu) and a copy of the modified files should be reported once modifications are completed, unless these modifications are made through the project's GitHub page: <http://www.csrc.sdsu.edu/mtk>. Documentation related to said modifications should be developed and included in any deliverable.
2. Redistributions of source code must be done through direct downloads from the project's GitHub page: <http://www.csrc.sdsu.edu/mtk>
3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. Usage of the binary form on proprietary applications shall require explicit prior written permission from the the copyright holders, and due credit should be given to the copyright holders.
5. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.



## Chapter 9

# Todo List

**Member `mtk::DenseMatrix::Kron` (`const DenseMatrix &aa, const DenseMatrix &bb`)**

Implement Kronecker product using the BLAS.

Implement Kron using the BLAS.

**Member `mtk::DenseMatrix::OrderColMajor` ()**

Improve this so that no new arrays have to be created.

**Member `mtk::DenseMatrix::OrderRowMajor` ()**

Improve this so that no new arrays have to be created.

**Member `mtk::DenseMatrix::Transpose` ()**

Improve this so that no extra arrays have to be created.

**Class `mtk::GLPKAdapter`**

Rescind from the GLPK as the numerical core for CLO problems.

**Member `mtk::Matrix::IncreaseNumNull` () noexcept**

Review the definition of sparse matrices properties.

**Member `mtk::Matrix::IncreaseNumZero` () noexcept**

Review the definition of sparse matrices properties.

**Member `mtk::RobinBCDescriptor2D::ImposeOnGrid` (`UniStgGrid2D &grid, const Real &time=kZero`) const**

Implement imposition for vector-valued grids. Need research here!

**Member `mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace` (`const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero`) const**

Impose the Neumann conditions on every pole, for every scenario.

**Member `mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace` (`const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero`) const**

Impose Harmonic mean on the corners for the case when the generated space is available, for all poles.

**Member `mtk::Tools::Prevent` (`const bool complement, const char *const fname, int lineno, const char *const fxname`) noexcept**

Check if this is the best way of stalling execution.

**Member `mtk::UniStgGrid1D::discrete_domain_x` () const**

Review const-correctness of the pointer we return.

**Member `mtk::UniStgGrid1D::discrete_field` ()**

Review const-correctness of the pointer we return. Look at the STL!

**Member [mtk::UniStgGrid2D::discrete\\_domain\\_x \(\) const](#)**

Review const-correctness of the pointer we return.

**Member [mtk::UniStgGrid2D::discrete\\_domain\\_y \(\) const](#)**

Review const-correctness of the pointer we return.

**Member [mtk::UniStgGrid3D::discrete\\_domain\\_x \(\) const](#)**

Review const-correctness of the pointer we return.

**Member [mtk::UniStgGrid3D::discrete\\_domain\\_y \(\) const](#)**

Review const-correctness of the pointer we return.

**Member [mtk::UniStgGrid3D::discrete\\_domain\\_z \(\) const](#)**

Review const-correctness of the pointer we return.

**File [mtk\\_blas\\_adapter.cc](#)**

Write documentation using LaTeX.

**File [mtk\\_div\\_1d.cc](#)**

Overload ostream operator as in [mtk::Lap1D](#).

Implement creation of ■ w. [mtk::BLASAdapter](#).

**File [mtk\\_glpk\\_adapter\\_test.cc](#)**

Test the [mtk::GLPKAdapter](#) class.

**File [mtk\\_grad\\_1d.cc](#)**

Overload ostream operator as in [mtk::Lap1D](#).

Implement creation of ■ w. [mtk::BLASAdapter](#).

**File [mtk\\_lapack\\_adapter.cc](#)**

Write documentation using LaTeX.

**File [mtk\\_lapack\\_adapter\\_test.cc](#)**

Test the [mtk::LAPACKAdapter](#) class.

**File [mtk\\_quad\\_1d.h](#)**

Implement this class.

**File [mtk\\_roots.h](#)**

Test selective precision mechanisms.

**File [mtk\\_uni\\_stg\\_grid\\_1d.h](#)**

Create overloaded binding routines that read data from files.

**File [mtk\\_uni\\_stg\\_grid\\_2d.h](#)**

Create overloaded binding routines that read data from files.

**File [mtk\\_uni\\_stg\\_grid\\_3d.h](#)**

Create overloaded binding routines that read data from files.



## Chapter 10

# Bug List

**Member `mtk::Matrix::set_num_null` (`const int &in`) `noexcept`**

-nan assigned on construction time due to `num_values_` being 0.

**Member `mtk::Matrix::set_num_zero` (`const int &in`) `noexcept`**

-nan assigned on construction time due to `num_values_` being 0.



# Chapter 11

## Module Index

### 11.1 Modules

Here is a list of all modules:

Roots. . . . .	33
Enumerations. . . . .	36
Execution tools. . . . .	38
Data structures. . . . .	39
Numerical methods. . . . .	40
Grids. . . . .	41
Mimetic operators. . . . .	42



## Chapter 12

# Namespace Index

### 12.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">mtk</a>	Mimetic Methods Toolkit namespace . . . . .	<a href="#">45</a>
---------------------	---------------------------------------------	--------------------



## Chapter 13

# Class Index

### 13.1 Class List

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

<a href="#">mtk::BLASAdapter</a>	Adapter class for the BLAS API . . . . .	55
<a href="#">mtk::Curl2D</a>	Implements a 2D mimetic curl operator . . . . .	63
<a href="#">mtk::DenseMatrix</a>	Defines a common dense matrix, using a 1D array . . . . .	67
<a href="#">mtk::Div1D</a>	Implements a 1D mimetic divergence operator . . . . .	86
<a href="#">mtk::Div2D</a>	Implements a 2D mimetic divergence operator . . . . .	98
<a href="#">mtk::Div3D</a>	Implements a 3D mimetic divergence operator . . . . .	102
<a href="#">mtk::GLPKAdapter</a>	Adapter class for the GLPK API . . . . .	107
<a href="#">mtk::Grad1D</a>	Implements a 1D mimetic gradient operator . . . . .	110
<a href="#">mtk::Grad2D</a>	Implements a 2D mimetic gradient operator . . . . .	123
<a href="#">mtk::Grad3D</a>	Implements a 3D mimetic gradient operator . . . . .	127
<a href="#">mtk::Interp1D</a>	Implements a 1D interpolation operator . . . . .	132
<a href="#">mtk::Interp2D</a>	Implements a 2D interpolation operator . . . . .	135
<a href="#">mtk::Lap1D</a>	Implements a 1D mimetic Laplacian operator . . . . .	138
<a href="#">mtk::Lap2D</a>	Implements a 2D mimetic Laplacian operator . . . . .	145
<a href="#">mtk::Lap3D</a>	Implements a 3D mimetic Laplacian operator . . . . .	149
<a href="#">mtk::LAPACKAdapter</a>	Adapter class for the LAPACK API . . . . .	153
<a href="#">mtk::Matrix</a>	Definition of the representation of a matrix in the MTK . . . . .	161

<a href="#">mtk::Quad1D</a>	Implements a 1D mimetic quadrature . . . . .	177
<a href="#">mtk::RobinBCDescriptor1D</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	179
<a href="#">mtk::RobinBCDescriptor2D</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	187
<a href="#">mtk::RobinBCDescriptor3D</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	206
<a href="#">mtk::Tools</a>	Tool manager class . . . . .	215
<a href="#">mtk::UniStgGrid1D</a>	Uniform 1D Staggered Grid . . . . .	218
<a href="#">mtk::UniStgGrid2D</a>	Uniform 2D Staggered Grid . . . . .	225
<a href="#">mtk::UniStgGrid3D</a>	Uniform 3D Staggered Grid . . . . .	242



## Chapter 14

# File Index

### 14.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Makefile.inc</a>	356
examples/curl_2d/angular_velocity/ <a href="#">curl_2d_angular_velocity.cc</a>	
Compute the curl of a 2D angular velocity field	257
examples/diffusion_3d/ <a href="#">diffusion_3d.cc</a>	
Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs	259
examples/divergence_operators_1d/ <a href="#">divergence_operators_1d.cc</a>	
Creates instances of a 1D divergence as computed by the CBS algorithm	262
examples/divergence_operators_1d_mimetic_test/ <a href="#">divergence_operators_1d_mimetic_test.cc</a>	
Test mimetic qualities of instances of a 1D divergence from the CBSA	264
examples/minimalistic_poisson_1d/ <a href="#">minimalistic_poisson_1d.cc</a>	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	266
examples/poisson_1d/ <a href="#">poisson_1d.cc</a>	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	269
examples/poisson_2d/ <a href="#">poisson_2d.cc</a>	
Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs	273
examples/pos_def_weights_1d/ <a href="#">pos_def_weights_1d.cc</a>	
The CBS algorithm computes positive-definite weights, for 1D operators	277
include/ <a href="#">mtk.h</a>	
Includes the entire API	279
include/ <a href="#">mtk_blas_adapter.h</a>	
Adapter class for the BLAS API	282
include/ <a href="#">mtk_curl_2d.h</a>	
Includes the definition of the class Curl2D	284
include/ <a href="#">mtk_dense_matrix.h</a>	
Defines a common dense matrix, using a 1D array	287
include/ <a href="#">mtk_div_1d.h</a>	
Includes the definition of the class Div1D	290
include/ <a href="#">mtk_div_2d.h</a>	
Includes the definition of the class Div2D	292
include/ <a href="#">mtk_div_3d.h</a>	
Includes the definition of the class Div3D	295
include/ <a href="#">mtk_enums.h</a>	
Considered enumeration types in the MTK	297

include/ <a href="#">mtk_glpk_adapter.h</a>	Adapter class for the GLPK API . . . . .	299
include/ <a href="#">mtk_grad_1d.h</a>	Includes the definition of the class Grad1D . . . . .	302
include/ <a href="#">mtk_grad_2d.h</a>	Includes the definition of the class Grad2D . . . . .	305
include/ <a href="#">mtk_grad_3d.h</a>	Includes the definition of the class Grad3D . . . . .	307
include/ <a href="#">mtk_interp_1d.h</a>	Includes the definition of the class Interp1D . . . . .	310
include/ <a href="#">mtk_interp_2d.h</a>	Includes the definition of the class Interp2D . . . . .	313
include/ <a href="#">mtk_lap_1d.h</a>	Includes the definition of the class Lap1D . . . . .	315
include/ <a href="#">mtk_lap_2d.h</a>	Includes the implementation of the class Lap2D . . . . .	318
include/ <a href="#">mtk_lap_3d.h</a>	Includes the implementation of the class Lap3D . . . . .	320
include/ <a href="#">mtk_lapack_adapter.h</a>	Adapter class for the LAPACK API . . . . .	323
include/ <a href="#">mtk_matrix.h</a>	Definition of the representation of a matrix in the MTK . . . . .	326
include/ <a href="#">mtk_quad_1d.h</a>	Includes the definition of the class Quad1D . . . . .	328
include/ <a href="#">mtk_robin_bc_descriptor_1d.h</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	331
include/ <a href="#">mtk_robin_bc_descriptor_2d.h</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	334
include/ <a href="#">mtk_robin_bc_descriptor_3d.h</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	338
include/ <a href="#">mtk_roots.h</a>	Fundamental definitions to be used across all classes of the MTK . . . . .	342
include/ <a href="#">mtk_tools.h</a>	Tool manager class . . . . .	345
include/ <a href="#">mtk_uni_stg_grid_1d.h</a>	Definition of an 1D uniform staggered grid . . . . .	347
include/ <a href="#">mtk_uni_stg_grid_2d.h</a>	Definition of an 2D uniform staggered grid . . . . .	349
include/ <a href="#">mtk_uni_stg_grid_3d.h</a>	Definition of an 3D uniform staggered grid . . . . .	352
src/ <a href="#">mtk_blas_adapter.cc</a>	Adapter class for the BLAS API . . . . .	361
src/ <a href="#">mtk_curl_2d.cc</a>	Implements the class Curl2D . . . . .	367
src/ <a href="#">mtk_dense_matrix.cc</a>	. . . . .	370
src/ <a href="#">mtk_div_1d.cc</a>	Implements the class Div1D . . . . .	377
src/ <a href="#">mtk_div_2d.cc</a>	Implements the class Div2D . . . . .	397
src/ <a href="#">mtk_div_3d.cc</a>	Implements the class Div3D . . . . .	399
src/ <a href="#">mtk_glpk_adapter.cc</a>	Adapter class for the GLPK API . . . . .	403

<a href="#">src/mtk_grad_1d.cc</a>	Implements the class Grad1D . . . . .	407
<a href="#">src/mtk_grad_2d.cc</a>	Implements the class Grad2D . . . . .	428
<a href="#">src/mtk_grad_3d.cc</a>	Implements the class Grad3D . . . . .	430
<a href="#">src/mtk_interp_1d.cc</a>	Includes the implementation of the class Interp1D . . . . .	433
<a href="#">src/mtk_lap_1d.cc</a>	Includes the implementation of the class Lap1D . . . . .	437
<a href="#">src/mtk_lap_2d.cc</a>	Includes the implementation of the class Lap2D . . . . .	442
<a href="#">src/mtk_lap_3d.cc</a>	Includes the implementation of the class Lap3D . . . . .	444
<a href="#">src/mtk_lapack_adapter.cc</a>	Adapter class for the LAPACK API . . . . .	446
<a href="#">src/mtk_matrix.cc</a>	Implementing the representation of a matrix in the MTK . . . . .	455
<a href="#">src/mtk_robin_bc_descriptor_1d.cc</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	460
<a href="#">src/mtk_robin_bc_descriptor_2d.cc</a>	Impose Robin boundary conditions on the operators and on the grids . . . . .	463
<a href="#">src/mtk_tools.cc</a>	Tool manager class . . . . .	474
<a href="#">src/mtk_uni_stg_grid_1d.cc</a>	Implementation of an 1D uniform staggered grid . . . . .	476
<a href="#">src/mtk_uni_stg_grid_2d.cc</a>	Implementation of a 2D uniform staggered grid . . . . .	480
<a href="#">src/mtk_uni_stg_grid_3d.cc</a>	Implementation of a 2D uniform staggered grid . . . . .	487
<a href="#">tests/mtk_blas_adapter_test.cc</a>	Test file for the <a href="#">mtk::BLASAdapter</a> class . . . . .	494
<a href="#">tests/mtk_dense_matrix_test.cc</a>	Test file for the <a href="#">mtk::DenseMatrix</a> class . . . . .	496
<a href="#">tests/mtk_div_1d_test.cc</a>	Testing the mimetic 1D divergence, constructed with the CBS algorithm . . . . .	501
<a href="#">tests/mtk_div_2d_test.cc</a>	Test file for the <a href="#">mtk::Div2D</a> class . . . . .	506
<a href="#">tests/mtk_div_3d_test.cc</a>	Test file for the <a href="#">mtk::Div3D</a> class . . . . .	508
<a href="#">tests/mtk_glpk_adapter_test.cc</a>	Test file for the <a href="#">mtk::GLPKAdapter</a> class . . . . .	511
<a href="#">tests/mtk_grad_1d_test.cc</a>	Testing the mimetic 1D gradient, constructed with the CBS algorithm . . . . .	513
<a href="#">tests/mtk_grad_2d_test.cc</a>	Test file for the <a href="#">mtk::Grad2D</a> class . . . . .	518
<a href="#">tests/mtk_grad_3d_test.cc</a>	Test file for the <a href="#">mtk::Grad3D</a> class . . . . .	520
<a href="#">tests/mtk_interp_1d_test.cc</a>	Testing the 1D interpolation . . . . .	523
<a href="#">tests/mtk_lap_1d_test.cc</a>	Testing the 1D Laplacian operator . . . . .	525
<a href="#">tests/mtk_lap_2d_test.cc</a>	Test file for the <a href="#">mtk::Lap2D</a> class . . . . .	529

tests/ <a href="#">mtk_lap_3d_test.cc</a>	
Test file for the <a href="#">mtk::Lap3D</a> class . . . . .	531
tests/ <a href="#">mtk_lapack_adapter_test.cc</a>	
Test file for the <a href="#">mtk::LAPACKAdapter</a> class . . . . .	534
tests/ <a href="#">mtk_robin_bc_descriptor_2d_test.cc</a>	
Test file for the <a href="#">mtk::RobinBCDescriptor2D</a> class . . . . .	536
tests/ <a href="#">mtk_uni_stg_grid_1d_test.cc</a>	
Test file for the <a href="#">mtk::UniStgGrid1D</a> class . . . . .	539
tests/ <a href="#">mtk_uni_stg_grid_2d_test.cc</a>	
Test file for the <a href="#">mtk::UniStgGrid2D</a> class . . . . .	543
tests/ <a href="#">mtk_uni_stg_grid_3d_test.cc</a>	
Test file for the <a href="#">mtk::UniStgGrid3D</a> class . . . . .	546

## Chapter 15

# Module Documentation

### 15.1 Roots.

Fundamental execution parameters and defined types.

#### Typedefs

- typedef float [mtk::Real](#)

*Users can simply change this to build a double- or single-precision MTK.*

#### Variables

- const float [mtk::kZero](#) {0.0f}  
*MTK's zero defined according to selective compilation.*
- const float [mtk::kOne](#) {1.0f}  
*MTK's one defined according to selective compilation.*
- const float [mtk::kTwo](#) {2.0f}  
*MTK's two defined according to selective compilation.*
- const float [mtk::kDefaultTolerance](#) {1e-7f}  
*Considered tolerance for comparisons in numerical methods.*
- const float [mtk::kDefaultMimeticThreshold](#) {1e-6f}  
*Default tolerance for higher-order mimetic operators.*
- const int [mtk::kDefaultOrderAccuracy](#) {2}  
*Default order of accuracy for mimetic operators.*
- const int [mtk::kCriticalOrderAccuracyGrad](#) {10}  
*At this order (and higher) we must use the CBSA to construct gradients.*
- const int [mtk::kCriticalOrderAccuracyDiv](#) {8}  
*At this order (and higher) we must use the CBSA to construct divergences.*

#### 15.1.1 Detailed Description

Fundamental execution parameters and defined types.

## 15.1.2 Typedef Documentation

### 15.1.2.1 `mtk::Real`

#### Warning

Defined as double if `MTK_PRECISION_DOUBLE` is defined on [Makefile.inc](#).

Definition at line 93 of file [mtk\\_roots.h](#).

## 15.1.3 Variable Documentation

### 15.1.3.1 `mtk::kCriticalOrderAccuracyDiv {8}`

Definition at line 186 of file [mtk\\_roots.h](#).

### 15.1.3.2 `mtk::kCriticalOrderAccuracyGrad {10}`

Definition at line 177 of file [mtk\\_roots.h](#).

### 15.1.3.3 `mtk::kDefaultMimeticThreshold {1e-6f}`

#### Warning

Declared as double if `MTK_PRECISION_DOUBLE` is defined on [Makefile.inc](#).

Definition at line 158 of file [mtk\\_roots.h](#).

### 15.1.3.4 `mtk::kDefaultOrderAccuracy {2}`

Definition at line 168 of file [mtk\\_roots.h](#).

### 15.1.3.5 `mtk::kDefaultTolerance {1e-7f}`

#### Warning

Declared as double if `MTK_PRECISION_DOUBLE` is defined on [Makefile.inc](#).

Definition at line 143 of file [mtk\\_roots.h](#).

### 15.1.3.6 `mtk::kOne {1.0f}`

#### Warning

Declared as double if `MTK_PRECISION_DOUBLE` is defined on [Makefile.inc](#).

Definition at line 127 of file [mtk\\_roots.h](#).

## 15.1.3.7 mtk::kTwo {2.0f}

## Warning

Declared as double if MTK\_PRECISION\_DOUBLE is defined on [Makefile.inc](#).

Definition at line 128 of file [mtk\\_roots.h](#).

## 15.1.3.8 mtk::kZero {0.0f}

## Warning

Declared as double if MTK\_PRECISION\_DOUBLE is defined on [Makefile.inc](#).

Definition at line 126 of file [mtk\\_roots.h](#).

## 15.2 Enumerations.

Enumerations.

### Enumerations

- enum `mtk::MatrixStorage` { `mtk::MatrixStorage::DENSE`, `mtk::MatrixStorage::BANDED`, `mtk::MatrixStorage::CRS` }  
*Considered matrix storage schemes to implement sparse matrices.*
- enum `mtk::MatrixOrdering` { `mtk::MatrixOrdering::ROW_MAJOR`, `mtk::MatrixOrdering::COL_MAJOR` }  
*Considered matrix ordering (for Fortran purposes).*
- enum `mtk::FieldNature` { `mtk::FieldNature::SCALAR`, `mtk::FieldNature::VECTOR` }  
*Nature of the field discretized in a given grid.*
- enum `mtk::DirInterp` { `mtk::DirInterp::SCALAR_TO_VECTOR`, `mtk::DirInterp::VECTOR_TO_SCALAR` }  
*Interpolation operator.*

### 15.2.1 Detailed Description

Enumerations.

### 15.2.2 Enumeration Type Documentation

#### 15.2.2.1 enum `mtk::DirInterp` [strong]

Used to tag different directions of interpolation supported.

Enumerator

**SCALAR\_TO\_VECTOR** Interpolations places scalar on vectors' location.

**VECTOR\_TO\_SCALAR** Interpolations places vectors on scalars' location.

Definition at line 127 of file `mtk_enums.h`.

#### 15.2.2.2 enum `mtk::FieldNature` [strong]

Fields can be **scalar** or **vector** in nature.

See also

[https://en.wikipedia.org/wiki/Scalar\\_field](https://en.wikipedia.org/wiki/Scalar_field)  
[https://en.wikipedia.org/wiki/Vector\\_field](https://en.wikipedia.org/wiki/Vector_field)

Enumerator

**SCALAR** Scalar-valued field.

**VECTOR** Vector-valued field.

Definition at line 113 of file `mtk_enums.h`.



### 15.2.2.3 `enum mtk::MatrixOrdering` [strong]

Row-major ordering is used for most application in C/C++. For Fortran purposes, the matrices must be listed in a column-major ordering.

See also

[https://en.wikipedia.org/wiki/Row-major\\_order](https://en.wikipedia.org/wiki/Row-major_order)

Enumerator

**ROW\_MAJOR** Row-major ordering (C/C++).

**COL\_MAJOR** Column-major ordering (Fortran).

Definition at line 95 of file [mtk\\_enums.h](#).

### 15.2.2.4 `enum mtk::MatrixStorage` [strong]

The considered sparse storage schemes are selected so that these are compatible with some of the most used mathematical APIs, as follows: DENSE and BANDED for [BLAS](#), [LAPACK](#), and [ScaLAPACK](#). Finally, CRS for [SuperLU](#).

Enumerator

**DENSE** Dense matrices, implemented as a 1D array: [DenseMatrix](#).

**BANDED** Banded matrices ala LAPACK and ScaLAPACK: Must be implemented.

**CRS** Compressed-Rows Storage: Must be implemented.

Definition at line 77 of file [mtk\\_enums.h](#).

## 15.3 Execution tools.

Tools to ensure execution correctness.

### Classes

- class `mtk::Tools`  
*Tool manager class.*

### 15.3.1 Detailed Description

Tools to ensure execution correctness.

## 15.4 Data structures.

Fundamental data structures.

### Classes

- class [mtk::DenseMatrix](#)  
*Defines a common dense matrix, using a 1D array.*
- class [mtk::Matrix](#)  
*Definition of the representation of a matrix in the MTK.*

### 15.4.1 Detailed Description

Fundamental data structures.

## 15.5 Numerical methods.

Adapter classes and auxiliary numerical methods.

### Classes

- class [mtk::BLASAdapter](#)  
*Adapter class for the BLAS API.*
- class [mtk::GLPKAdapter](#)  
*Adapter class for the GLPK API.*
- class [mtk::LAPACKAdapter](#)  
*Adapter class for the LAPACK API.*

### 15.5.1 Detailed Description

Adapter classes and auxiliary numerical methods.

## 15.6 Grids.

Uniform rectangular staggered grids.

### Classes

- class [mtk::UniStgGrid1D](#)  
*Uniform 1D Staggered Grid.*
- class [mtk::UniStgGrid2D](#)  
*Uniform 2D Staggered Grid.*
- class [mtk::UniStgGrid3D](#)  
*Uniform 3D Staggered Grid.*

### 15.6.1 Detailed Description

Uniform rectangular staggered grids.

## 15.7 Mimetic operators.

Mimetic operators.

### Classes

- class [mtk::Curl2D](#)  
*Implements a 2D mimetic curl operator.*
- class [mtk::Div1D](#)  
*Implements a 1D mimetic divergence operator.*
- class [mtk::Div2D](#)  
*Implements a 2D mimetic divergence operator.*
- class [mtk::Div3D](#)  
*Implements a 3D mimetic divergence operator.*
- class [mtk::Grad1D](#)  
*Implements a 1D mimetic gradient operator.*
- class [mtk::Grad2D](#)  
*Implements a 2D mimetic gradient operator.*
- class [mtk::Grad3D](#)  
*Implements a 3D mimetic gradient operator.*
- class [mtk::Interp1D](#)  
*Implements a 1D interpolation operator.*
- class [mtk::Interp2D](#)  
*Implements a 2D interpolation operator.*
- class [mtk::Lap1D](#)  
*Implements a 1D mimetic Laplacian operator.*
- class [mtk::Lap2D](#)  
*Implements a 2D mimetic Laplacian operator.*
- class [mtk::Lap3D](#)  
*Implements a 3D mimetic Laplacian operator.*
- class [mtk::Quad1D](#)  
*Implements a 1D mimetic quadrature.*
- class [mtk::RobinBCDescriptor1D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*
- class [mtk::RobinBCDescriptor2D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*
- class [mtk::RobinBCDescriptor3D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*

### Typedefs

- typedef Real(\* [mtk::CoefficientFunction0D](#) )(const Real &tt)  
*A function of a BC coefficient evaluated on a 0D domain and time.*
- typedef Real(\* [mtk::CoefficientFunction1D](#) )(const Real &xx, const Real &tt)  
*A function of a BC coefficient evaluated on a 1D domain and time.*
- typedef Real(\* [mtk::CoefficientFunction2D](#) )(const Real &xx, const Real &yy, const Real &tt)  
*A function of a BC coefficient evaluated on a 2D domain and time.*

### 15.7.1 Detailed Description

Mimetic operators.

### 15.7.2 Typedef Documentation

#### 15.7.2.1 `mtk::CoefficientFunction0D`

##### Warning

This definition implies that, for now, coefficients will depend on space and time, thus no extra parameters can influence their behavior. We will fix this soon enough.

Definition at line 111 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

#### 15.7.2.2 `mtk::CoefficientFunction1D`

Definition at line 97 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

#### 15.7.2.3 `mtk::CoefficientFunction2D`

Definition at line 97 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).





## Chapter 16

# Namespace Documentation

### 16.1 mtk Namespace Reference

Mimetic Methods Toolkit namespace.

#### Classes

- class [BLASAdapter](#)  
*Adapter class for the BLAS API.*
- class [Curl2D](#)  
*Implements a 2D mimetic curl operator.*
- class [DenseMatrix](#)  
*Defines a common dense matrix, using a 1D array.*
- class [Div1D](#)  
*Implements a 1D mimetic divergence operator.*
- class [Div2D](#)  
*Implements a 2D mimetic divergence operator.*
- class [Div3D](#)  
*Implements a 3D mimetic divergence operator.*
- class [GLPKAdapter](#)  
*Adapter class for the GLPK API.*
- class [Grad1D](#)  
*Implements a 1D mimetic gradient operator.*
- class [Grad2D](#)  
*Implements a 2D mimetic gradient operator.*
- class [Grad3D](#)  
*Implements a 3D mimetic gradient operator.*
- class [Interp1D](#)  
*Implements a 1D interpolation operator.*
- class [Interp2D](#)  
*Implements a 2D interpolation operator.*
- class [Lap1D](#)  
*Implements a 1D mimetic Laplacian operator.*

- class [Lap2D](#)  
*Implements a 2D mimetic Laplacian operator.*
- class [Lap3D](#)  
*Implements a 3D mimetic Laplacian operator.*
- class [LAPACKAdapter](#)  
*Adapter class for the LAPACK API.*
- class [Matrix](#)  
*Definition of the representation of a matrix in the MTK.*
- class [Quad1D](#)  
*Implements a 1D mimetic quadrature.*
- class [RobinBCDescriptor1D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*
- class [RobinBCDescriptor2D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*
- class [RobinBCDescriptor3D](#)  
*Impose Robin boundary conditions on the operators and on the grids.*
- class [Tools](#)  
*Tool manager class.*
- class [UniStgGrid1D](#)  
*Uniform 1D Staggered Grid.*
- class [UniStgGrid2D](#)  
*Uniform 2D Staggered Grid.*
- class [UniStgGrid3D](#)  
*Uniform 3D Staggered Grid.*

## Typedefs

- typedef [Real](#)(\* [CoefficientFunction0D](#) )(const [Real](#) &tt)  
*A function of a BC coefficient evaluated on a 0D domain and time.*
- typedef [Real](#)(\* [CoefficientFunction1D](#) )(const [Real](#) &xx, const [Real](#) &tt)  
*A function of a BC coefficient evaluated on a 1D domain and time.*
- typedef [Real](#)(\* [CoefficientFunction2D](#) )(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*A function of a BC coefficient evaluated on a 2D domain and time.*
- typedef float [Real](#)  
*Users can simply change this to build a double- or single-precision MTK.*

## Enumerations

- enum [MatrixStorage](#) { [MatrixStorage::DENSE](#), [MatrixStorage::BANDED](#), [MatrixStorage::CRS](#) }  
*Considered matrix storage schemes to implement sparse matrices.*
- enum [MatrixOrdering](#) { [MatrixOrdering::ROW\\_MAJOR](#), [MatrixOrdering::COL\\_MAJOR](#) }  
*Considered matrix ordering (for Fortran purposes).*
- enum [FieldNature](#) { [FieldNature::SCALAR](#), [FieldNature::VECTOR](#) }  
*Nature of the field discretized in a given grid.*
- enum [DirInterp](#) { [DirInterp::SCALAR\\_TO\\_VECTOR](#), [DirInterp::VECTOR\\_TO\\_SCALAR](#) }  
*Interpolation operator.*

## Functions

- float [snrm2\\_](#) (int \*n, float \*x, int \*incx)
- void [saxpy\\_](#) (int \*n, float \*sa, float \*sx, int \*incx, float \*sy, int \*incy)
- void [sgemv\\_](#) (char \*trans, int \*m, int \*n, float \*alpha, float \*a, int \*lda, float \*x, int \*incx, float \*beta, float \*y, int \*incy)
- void [sgemm\\_](#) (char \*transa, char \*transb, int \*m, int \*n, int \*k, double \*alpha, double \*a, int \*lda, double \*b, aamm int \*ldb, double \*beta, double \*c, int \*ldc)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::DenseMatrix](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Div1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Grad1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Interp1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Lap1D](#) &in)
- void [sgesv\\_](#) (int \*n, int \*nrhs, [Real](#) \*a, int \*lda, int \*ipiv, [Real](#) \*b, int \*ldb, int \*info)
- void [sgels\\_](#) (char \*trans, int \*m, int \*n, int \*nrhs, [Real](#) \*a, int \*lda, [Real](#) \*b, int \*ldb, [Real](#) \*work, int \*lwork, int \*info)  
*Single-precision GEneral matrix Least Squares solver.*
- void [sgeqrf\\_](#) (int \*m, int \*n, [Real](#) \*a, int \*lda, [Real](#) \*tau, [Real](#) \*work, int \*lwork, int \*info)  
*Single-precision GEneral matrix QR Factorization.*
- void [sormqr\\_](#) (char \*side, char \*trans, int \*m, int \*n, int \*k, [Real](#) \*a, int \*lda, [Real](#) \*tau, [Real](#) \*c, int \*ldc, [Real](#) \*work, int \*lwork, int \*info)  
*Single-precision Orthogonal [Matrix](#) from QR factorization.*
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid2D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid3D](#) &in)

## Variables

- const float [kZero](#) {0.0f}  
*MTK's zero defined according to selective compilation.*
- const float [kOne](#) {1.0f}  
*MTK's one defined according to selective compilation.*
- const float [kTwo](#) {2.0f}  
*MTK's two defined according to selective compilation.*
- const float [kDefaultTolerance](#) {1e-7f}  
*Considered tolerance for comparisons in numerical methods.*
- const float [kDefaultMimeticThreshold](#) {1e-6f}  
*Default tolerance for higher-order mimetic operators.*
- const int [kDefaultOrderAccuracy](#) {2}  
*Default order of accuracy for mimetic operators.*
- const int [kCriticalOrderAccuracyGrad](#) {10}  
*At this order (and higher) we must use the CBSA to construct gradients.*
- const int [kCriticalOrderAccuracyDiv](#) {8}  
*At this order (and higher) we must use the CBSA to construct divergences.*

### 16.1.1 Function Documentation

16.1.1.1 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::Interp1D & in )`

1. Print approximating coefficients for the interior.

Definition at line 66 of file [mtk\\_interp\\_1d.cc](#).

16.1.1.2 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::UniStgGrid3D & in )`

1. Print spatial coordinates.
2. Print scalar field.

Definition at line 67 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

16.1.1.3 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::UniStgGrid2D & in )`

1. Print spatial coordinates.
2. Print scalar field.

Definition at line 67 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

16.1.1.4 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::UniStgGrid1D & in )`

1. Print spatial coordinates.
2. Print scalar field.

Definition at line 68 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

16.1.1.5 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::Lap1D & in )`

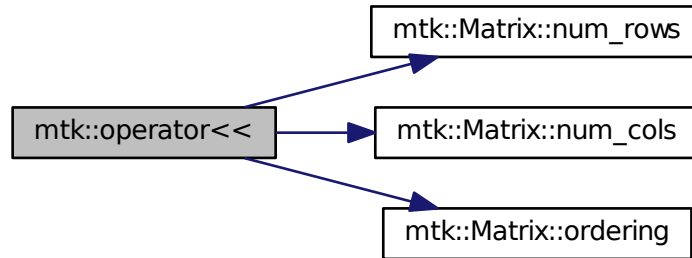
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

Definition at line 73 of file [mtk\\_lap\\_1d.cc](#).

16.1.1.6 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::DenseMatrix & in )`

Definition at line 79 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



16.1.1.7 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::Grad1D & in )`

1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

Definition at line 84 of file [mtk\\_grad\\_1d.cc](#).

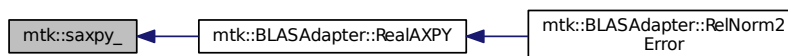
16.1.1.8 `std::ostream& mtk::operator<< ( std::ostream & stream, mtk::Div1D & in )`

1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

Definition at line 84 of file [mtk\\_div\\_1d.cc](#).

16.1.1.9 `void mtk::saxpy_ ( int * n, float * sa, float * sx, int * incx, float * sy, int * incy )`

Here is the caller graph for this function:



16.1.1.10 void mtk::sgels\_ ( char \* *trans*, int \* *m*, int \* *n*, int \* *nrhs*, Real \* *a*, int \* *lda*, Real \* *b*, int \* *ldb*, Real \* *work*, int \* *lwork*, int \* *info* )

SGELS solves overdetermined or underdetermined real linear systems involving an M-by-N matrix A, or its transpose, using a QR or LQ factorization of A. It is assumed that A has full rank.

The following options are provided:

1. If TRANS = 'N' and  $m \geq n$ : find the least squares solution of an overdetermined system, i.e., solve the least squares problem

$$\text{minimize } || B - A * X ||.$$

2. If TRANS = 'N' and  $m < n$ : find the minimum norm solution of an underdetermined system  $A * X = B$ .

3. If TRANS = 'T' and  $m \geq n$ : find the minimum norm solution of an undetermined system  $A^{**T} * X = B$ .

4. If TRANS = 'T' and  $m < n$ : find the least squares solution of an overdetermined system, i.e., solve the least squares problem

$$\text{minimize } || B - A^{**T} * X ||.$$

Several right hand side vectors b and solution vectors x can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

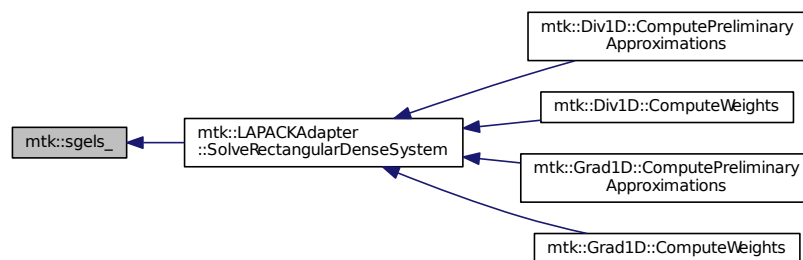
See also

<http://www.math.utah.edu/software/lapack/lapack-s/sgels.html>

#### Parameters

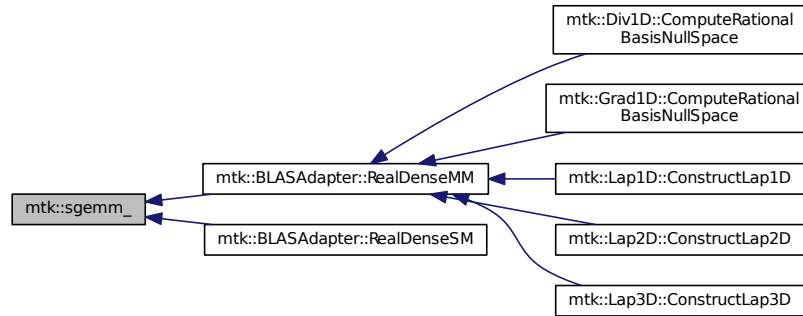
in	<i>trans</i>	Am I giving the transpose of the matrix?
in	<i>m</i>	The number of rows of the matrix a. $m \geq 0$ .
in	<i>n</i>	The number of columns of the matrix a. $n \geq 0$ .
in	<i>nrhs</i>	The number of right-hand sides.
in,out	<i>a</i>	On entry, the m-by-n matrix a.
in	<i>lda</i>	The leading dimension of a. $lda \geq \max(1,m)$ .
in,out	<i>b</i>	On entry, matrix b of right-hand side vectors.
in	<i>ldb</i>	The leading dimension of b. $ldb \geq \max(1,m,n)$ .
in,out	<i>work</i>	On exit, if <i>info</i> = 0, <i>work</i> (1) is optimal <i>lwork</i> .
in,out	<i>lwork</i>	The dimension of the array work.
in,out	<i>info</i>	If <i>info</i> = 0, then successful exit.

Here is the caller graph for this function:



16.1.1.11 void mtk::sgemm\_ ( char \* *transa*, char \* *transb*, int \* *m*, int \* *n*, int \* *k*, double \* *alpha*, double \* *a*, int \* *lda*, double \* *b*, aamm int \* *ldb*, double \* *beta*, double \* *c*, int \* *ldc* )

Here is the caller graph for this function:



16.1.1.12 void mtk::sgemv\_ ( char \* *trans*, int \* *m*, int \* *n*, float \* *alpha*, float \* *a*, int \* *lda*, float \* *x*, int \* *incx*, float \* *beta*, float \* *y*, int \* *incy* )

Here is the caller graph for this function:



16.1.1.13 void mtk::sgeqrf\_ ( int \* *m*, int \* *n*, Real \* *a*, int \* *lda*, Real \* *tau*, Real \* *work*, int \* *lwork*, int \* *info* )

Single-Precision Orthogonal Make Q from QR: dormqr\_ overwrites the general real M-by-N matrix C with (Table 1):

SIDE = 'L'      SIDE = 'R'

TRANS = 'N':  $Q * C * Q^T$  TRANS = 'T':  $Q^{*T} * C * Q^{*T}$

where Q is a real orthogonal matrix defined as the product of k elementary reflectors

$$Q = H(1) H(2) \dots H(k)$$

as returned by SGEQRF. Q is of order M if SIDE = 'L' and of order N if SIDE = 'R'.

See also

[http://www.netlib.org/lapack/explore-html/df/d97/sgeqrf\\_8f.html](http://www.netlib.org/lapack/explore-html/df/d97/sgeqrf_8f.html)

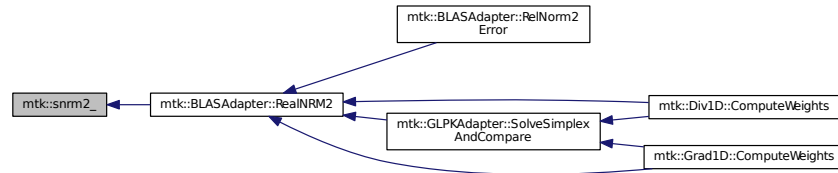
## Parameters

in	<i>m</i>	The number of columns of the matrix a. $n \geq 0$ .
in	<i>n</i>	The number of columns of the matrix a. $n \geq 0$ .
in, out	<i>a</i>	On entry, the n-by-n matrix a.
in	<i>lda</i>	Leading dimension matrix. $LDA \geq \max(1, M)$ .
in, out	<i>tau</i>	Scalars from elementary reflectors. $\min(M, N)$ .
in, out	<i>work</i>	Workspace. $info = 0$ , $work(1)$ is optimal $lwork$ .
in	<i>lwork</i>	The dimension of work. $lwork \geq \max(1, n)$ .
in	<i>info</i>	$info = 0$ : successful exit.

16.1.1.14 void mtk::sgesv\_( int \* n, int \* nrhs, Real \* a, int \* lda, int \* ipiv, Real \* b, int \* ldb, int \* info )

16.1.1.15 float mtk::snrm2\_( int \* n, float \* x, int \* incx )

Here is the caller graph for this function:



16.1.1.16 void mtk::sormqr\_( char \* side, char \* trans, int \* m, int \* n, int \* k, Real \* a, int \* lda, Real \* tau, Real \* c, int \* ldc, Real \* work, int \* lwork, int \* info )

Single-Precision Orthogonal Make Q from QR: sormqr\_ overwrites the general real M-by-N matrix C with (Table 1):

SIDE = 'L'      SIDE = 'R'

TRANS = 'N':  $Q * C * Q^T$  TRANS = 'T':  $Q^{*T} * C * Q^{*T}$

where Q is a real orthogonal matrix defined as the product of k elementary reflectors

$$Q = H(1) H(2) \dots H(k)$$

as returned by SGEQRF. Q is of order M if SIDE = 'L' and of order N if SIDE = 'R'.

## See also

[http://www.netlib.org/lapack/explore-html/d0/d98/sormqr\\_8f\\_source.html](http://www.netlib.org/lapack/explore-html/d0/d98/sormqr_8f_source.html)

## Parameters



in	<i>side</i>	See Table 1 above.
in	<i>trans</i>	See Table 1 above.
in	<i>m</i>	Number of rows of the C matrix.
in	<i>n</i>	Number of columns of the C matrix.
in	<i>k</i>	Number of reflectors.
in,out	<i>a</i>	The matrix containing the reflectors.
in	<i>lda</i>	The dimension of work. $lwork \geq \max(1,n)$ .
in	<i>tau</i>	Scalar factors of the elementary reflectors.
in	<i>c</i>	Output matrix.
in	<i>ldc</i>	Leading dimension of the output matrix.
in,out	<i>work</i>	Workspace. $info = 0$ , $work(1)$ optimal $lwork$ .
in	<i>lwork</i>	The dimension of work.
in,out	<i>info</i>	$info = 0$ : successful exit.



## Chapter 17

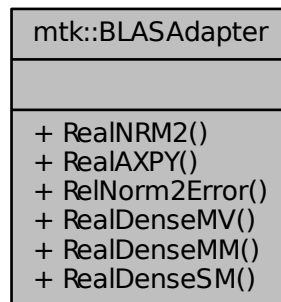
# Class Documentation

### 17.1 mtk::BLASAdapter Class Reference

Adapter class for the BLAS API.

```
#include <mtk_blas_adapter.h>
```

Collaboration diagram for mtk::BLASAdapter:



#### Static Public Member Functions

- static `Real RealNRM2 (Real *in, int &in_length)`  
*Compute the  $\|x\|_2$  of given array `x`.*
- static void `RealAXPY (Real alpha, Real *xx, Real *yy, int &in_length)`  
*Real-Arithmetic Scalar-Vector plus a Vector.*
- static `Real RelNorm2Error (Real *computed, Real *known, int length)`  
*Computes the relative norm-2 of the error.*
- static void `RealDenseMV (Real &alpha, DenseMatrix &aa, Real *xx, Real &beta, Real *yy)`  
*Real-Arithmetic General (Dense matrices) Matrix-Vector Multiplier.*

- static `DenseMatrix RealDenseMM (DenseMatrix &aa, DenseMatrix &bb)`

*Real-Arithmetic General (Dense matrices) Matrix-Matrix multiplier.*

- static `DenseMatrix RealDenseSM (Real alpha, DenseMatrix &aa)`

*Real-Arithmetic General (Dense matrices) Scalar-Matrix multiplier.*

### 17.1.1 Detailed Description

This class contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the BLAS.

The **BLAS (Basic Linear Algebra Subprograms)** are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.

See also

<http://www.netlib.org/blas/>  
<https://software.intel.com/en-us/non-commercial-software-development>

Definition at line 99 of file `mtk_blas_adapter.h`.

### 17.1.2 Member Function Documentation

17.1.2.1 `void mtk::BLASAdapter::RealAXPY ( mtk::Real alpha, mtk::Real * xx, mtk::Real * yy, int & in_length )`  
`[static]`

Performs

$$\mathbf{y} := \alpha \mathbf{A} \mathbf{x} + \mathbf{y}$$

Parameters

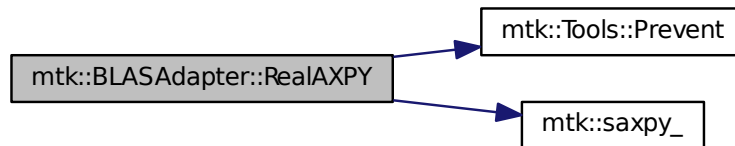
<code>in</code>	<code>alpha</code>	Scalar of the first array.
<code>in</code>	<code>xx</code>	First array.
<code>in</code>	<code>yy</code>	Second array.
<code>in</code>	<code>in_length</code>	Lengths of the given arrays.

**Returns**

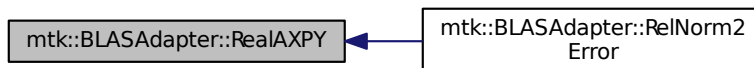
Norm-2 of the given array.

Definition at line 342 of file [mtk\\_blas\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.1.2.2 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM ( mtk::DenseMatrix & aa, mtk::DenseMatrix & bb ) [static]

Performs:

$$\mathbf{C} := \mathbf{AB}$$

**Parameters**

in	<i>aa</i>	First matrix.
in	<i>bb</i>	Second matrix.

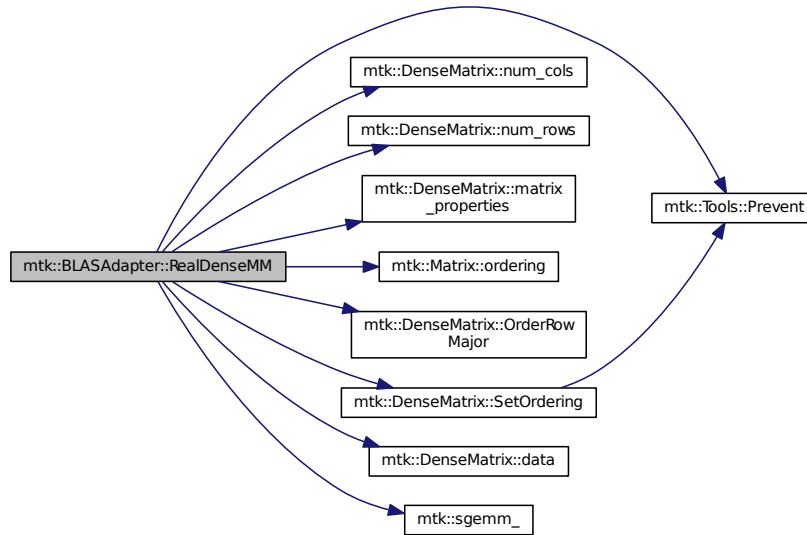
**See also**

<http://ejspeiro.github.io/Netlib-and-CPP/>

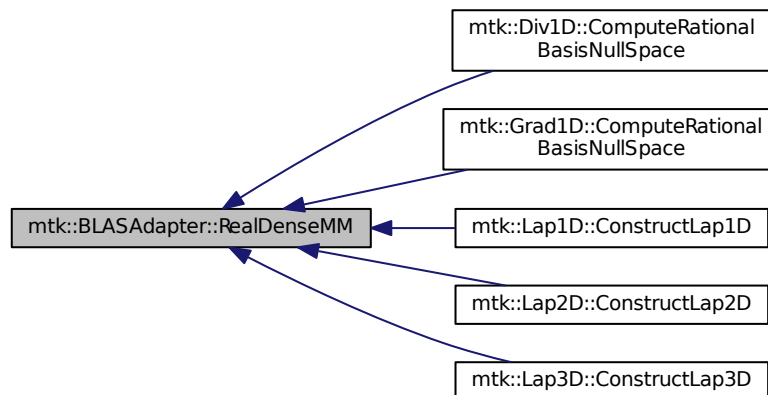
1. Make sure input matrices are row-major ordered.
2. Setup the problem.
3. Perform multiplication.

Definition at line 412 of file [mtk\\_blas\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.1.2.3 `void mtk::BLASAdapter::RealDenseMV ( mtk::Real & alpha, mtk::DenseMatrix & aa, mtk::Real * xx, mtk::Real & beta, mtk::Real * yy ) [static]`

Performs

$$\mathbf{y} := \alpha \mathbf{A}\mathbf{x} + \beta \mathbf{y}$$

**Parameters**

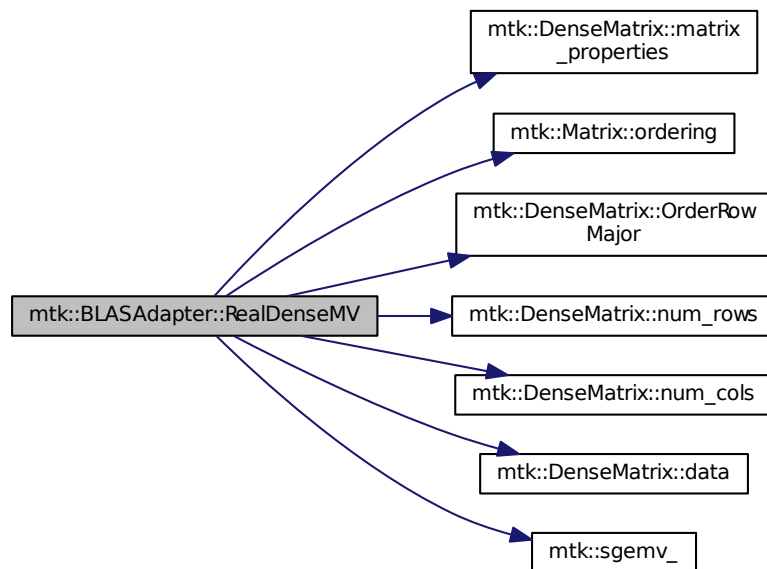
in	<i>alpha</i>	First scalar.
in	<i>aa</i>	Given matrix.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).

**See also**

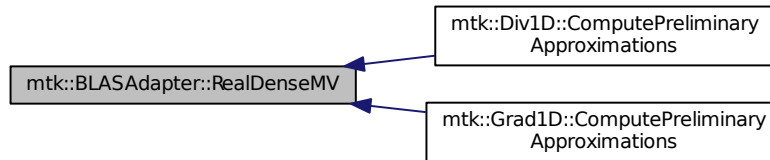
<http://ejspeiro.github.io/Netlib-and-CPP/>

Definition at line 381 of file `mtk_blas_adapter.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



17.1.2.4 `mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM ( mtk::Real alpha, mtk::DenseMatrix & aa ) [static]`

Performs:

$$\mathbf{B} := \alpha \mathbf{A}$$

#### Parameters

in	<i>alpha</i>	Input scalar.
in	<i>aa</i>	Input matrix.

#### See also

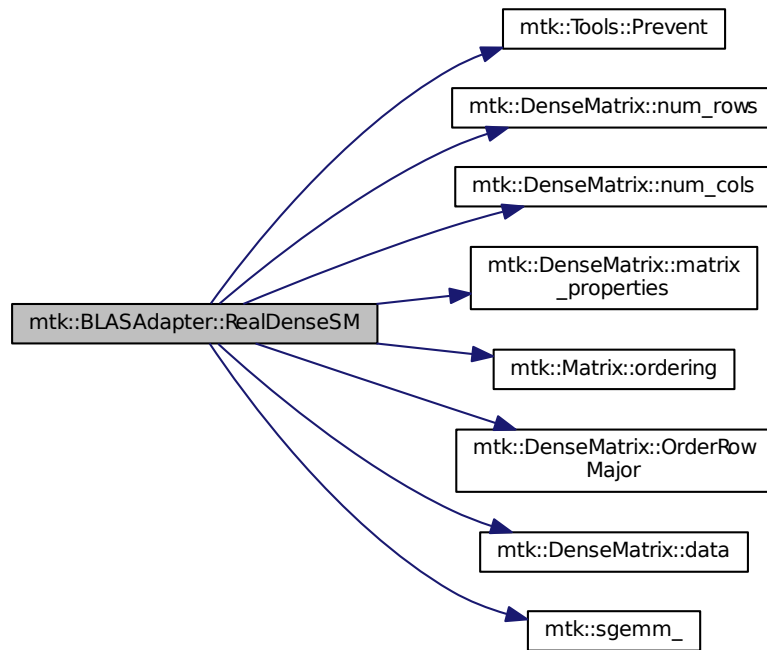
<http://ejspeiro.github.io/Netlib-and-CPP/>

1. Make sure input matrices are row-major ordered.
2. Setup the problem.
3. Perform multiplication.

Definition at line 469 of file `mtk_blas_adapter.cc`.



Here is the call graph for this function:



**17.1.2.5** `mtk::Real mtk::BLASAdapter::RealNRM2 ( Real * in, int & in_length ) [static]`

#### Parameters

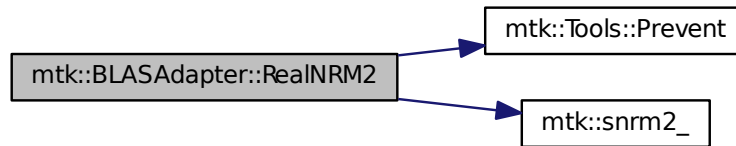
<code>in</code>	<code>in</code>	Input array.
<code>in</code>	<code>in_length</code>	Length of the array.

**Returns**

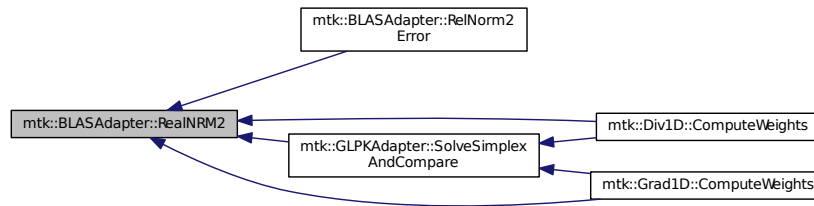
Norm-2 of the given array.

Definition at line 327 of file [mtk\\_blas\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**17.1.2.6** `mtk::Real mtk::BLASAdapter::RelNorm2Error ( mtk::Real * computed, mtk::Real * known, int length )`  
`[static]`

We compute

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2}{\|\mathbf{x}\|_2}.$$

**Parameters**

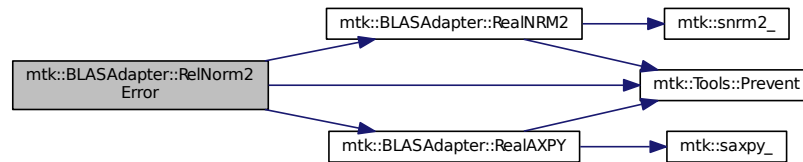
<i>in</i>	<i>known</i>	Array containing the computed solution.
<i>in</i>	<i>computed</i>	Array containing the known solution (ref. solution).

**Returns**

Relative norm-2 of the error, aka, the difference between the arrays.

Definition at line 361 of file [mtk\\_blas\\_adapter.cc](#).

Here is the call graph for this function:



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

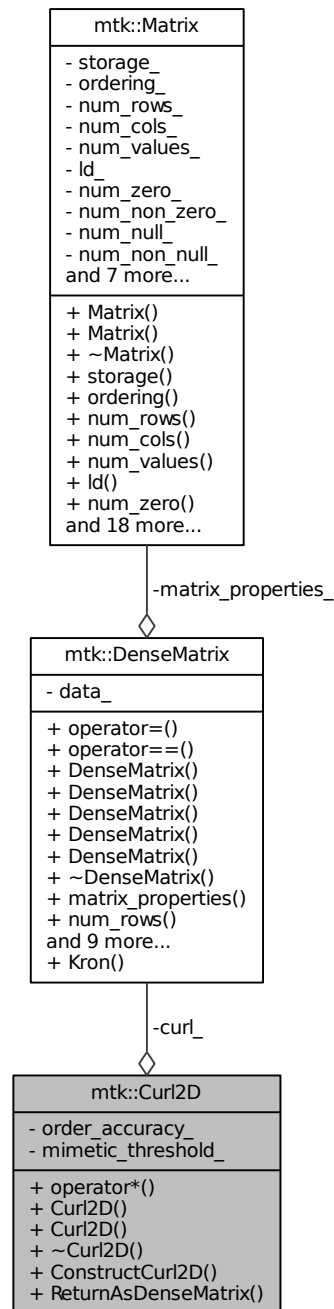
- [include/mtk\\_blas\\_adapter.h](#)
- [src/mtk\\_blas\\_adapter.cc](#)

## 17.2 mtk::Curl2D Class Reference

Implements a 2D mimetic curl operator.

```
#include <mtk_curl_2d.h>
```

Collaboration diagram for mtk::Curl2D:



## Public Member Functions

- [UniStgGrid3D operator\\*](#) (const [UniStgGrid2D](#) &grid) const

*Operator application operator on a grid.*

- [Curl2D](#) ()

*Default constructor.*

- [Curl2D](#) (const [Curl2D](#) &curl)

*Copy constructor.*

- [~Curl2D](#) ()

*Destructor.*

- bool [ConstructCurl2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=kDefaultOrderAccuracy, [Real](#) mimetic\_↔ threshold=kDefaultMimeticThreshold)

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix](#) curl\_

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real](#) [mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.2.1 Detailed Description

This class implements a 2D curl operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 77 of file [mtk\\_curl\\_2d.h](#).

## 17.2.2 Constructor & Destructor Documentation

### 17.2.2.1 mtk::Curl2D::Curl2D ( )

Definition at line 79 of file [mtk\\_curl\\_2d.cc](#).

### 17.2.2.2 mtk::Curl2D::Curl2D ( const [Curl2D](#) &curl )

Parameters

<a href="#">in</a>	<a href="#">curl</a>	Given curl.
--------------------	----------------------	-------------

Definition at line 83 of file [mtk\\_curl\\_2d.cc](#).

### 17.2.2.3 mtk::Curl2D::~~Curl2D ( )

Definition at line 87 of file [mtk\\_curl\\_2d.cc](#).

### 17.2.3 Member Function Documentation

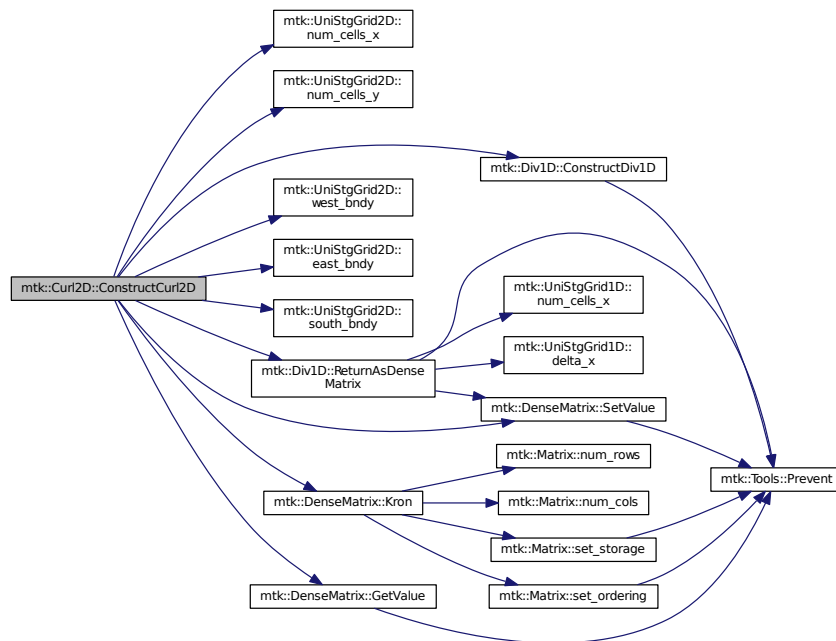
17.2.3.1 `bool mtk::Curl2D::ConstructCurl2D ( const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

Definition at line 89 of file [mtk\\_curl\\_2d.cc](#).

Here is the call graph for this function:



17.2.3.2 `mtk::UniStgGrid3D mtk::Curl2D::operator* ( const UniStgGrid2D & grid ) const`

1. Convert given vector field, into the required auxiliary vector field.

Definition at line 70 of file [mtk\\_curl\\_2d.cc](#).

17.2.3.3 `mtk::DenseMatrix mtk::Curl2D::ReturnAsDenseMatrix ( ) const`

#### Returns

The operator as a dense matrix.

Definition at line 157 of file [mtk\\_curl\\_2d.cc](#).

## 17.2.4 Member Data Documentation

### 17.2.4.1 DenseMatrix mtk::Curl2D::curl\_ [private]

Definition at line 112 of file [mtk\\_curl\\_2d.h](#).

### 17.2.4.2 Real mtk::Curl2D::mimetic\_threshold\_ [private]

Definition at line 116 of file [mtk\\_curl\\_2d.h](#).

### 17.2.4.3 int mtk::Curl2D::order\_accuracy\_ [private]

Definition at line 114 of file [mtk\\_curl\\_2d.h](#).

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

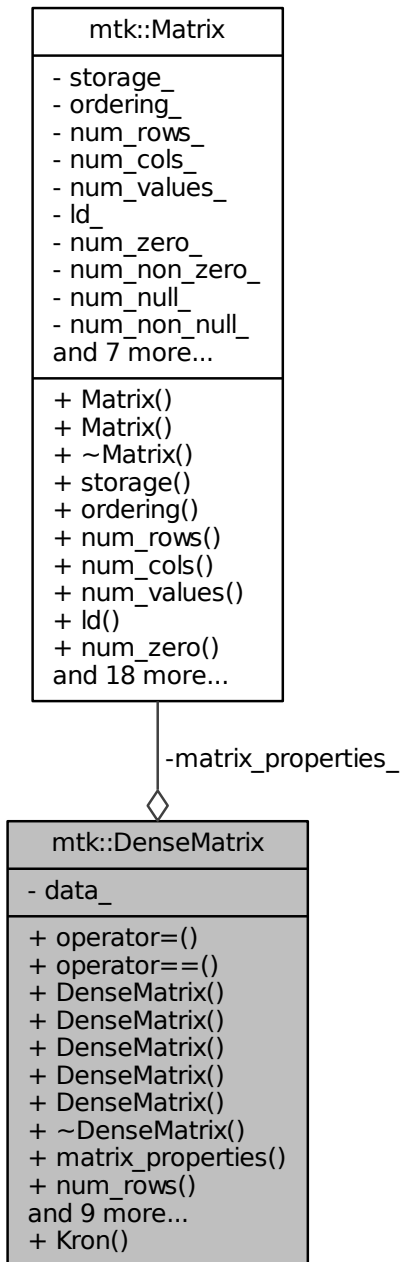
- [include/mtk\\_curl\\_2d.h](#)
- [src/mtk\\_curl\\_2d.cc](#)

## 17.3 mtk::DenseMatrix Class Reference

Defines a common dense matrix, using a 1D array.

```
#include <mtk_dense_matrix.h>
```

Collaboration diagram for mtk::DenseMatrix:



## Public Member Functions

- [DenseMatrix](#) & [operator=](#) (const [DenseMatrix](#) &in)



*Overloaded assignment operator.*

- `bool operator== (const DenseMatrix &in)`  
*Am I equal to the in matrix?*
- `DenseMatrix ()`  
*Default constructor.*
- `DenseMatrix (const DenseMatrix &in)`  
*Copy constructor.*
- `DenseMatrix (const int &num_rows, const int &num_cols)`  
*Construct a dense matrix based on the given dimensions.*
- `DenseMatrix (const int &rank, const bool &padded, const bool &transpose)`  
*Construct a zero-rows-padded identity matrix.*
- `DenseMatrix (const Real *const gen, const int &gen_length, const int &pro_length, const bool &transpose)`  
*Construct a dense Vandermonde matrix.*
- `~DenseMatrix ()`  
*Destructor.*
- `Matrix matrix_properties () const noexcept`  
*Provides access to the matrix data.*
- `int num_rows () const noexcept`  
*Gets the number of rows.*
- `int num_cols () const noexcept`  
*Gets the number of columns.*
- `Real * data () const noexcept`  
*Provides access to the matrix value array.*
- `void SetOrdering (mtk::MatrixOrdering oo) noexcept`  
*Sets the ordering of the matrix.*
- `Real GetValue (const int &row_coord, const int &col_coord) const noexcept`  
*Gets a value on the given coordinates.*
- `void SetValue (const int &row_coord, const int &col_coord, const Real &val) noexcept`  
*Sets a value on the given coordinates.*
- `void Transpose ()`  
*Transpose this matrix.*
- `void OrderRowMajor ()`  
*Make the matrix row-wise ordered.*
- `void OrderColMajor ()`  
*Make the matrix column-wise ordered.*
- `bool WriteToFile (const std::string &filename) const`  
*Writes matrix to a file compatible with Gnuplot 4.6.*

## Static Public Member Functions

- `static DenseMatrix Kron (const DenseMatrix &aa, const DenseMatrix &bb)`  
*Construct a dense matrix based on the Kronecker product of arguments.*

## Private Attributes

- [Matrix](#) `matrix_properties_`

*Data related to the matrix nature.*

- [Real](#) \* `data_`

*Array holding the data in contiguous position in memory.*

## Friends

- `std::ostream & operator<< (std::ostream &stream, DenseMatrix &in)`

*Prints the matrix as a block of numbers (standard way).*

### 17.3.1 Detailed Description

For developing purposes, it is better to have a not-so-intricated data structure implementing matrices. This is the purpose of this class: to be used for prototypes of new code for small test cases. In every other instance, this should be replaced by the most appropriate sparse matrix.

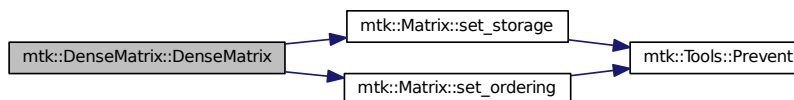
Definition at line 92 of file [mtk\\_dense\\_matrix.h](#).

### 17.3.2 Constructor & Destructor Documentation

#### 17.3.2.1 `mtk::DenseMatrix::DenseMatrix ( )`

Definition at line 167 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



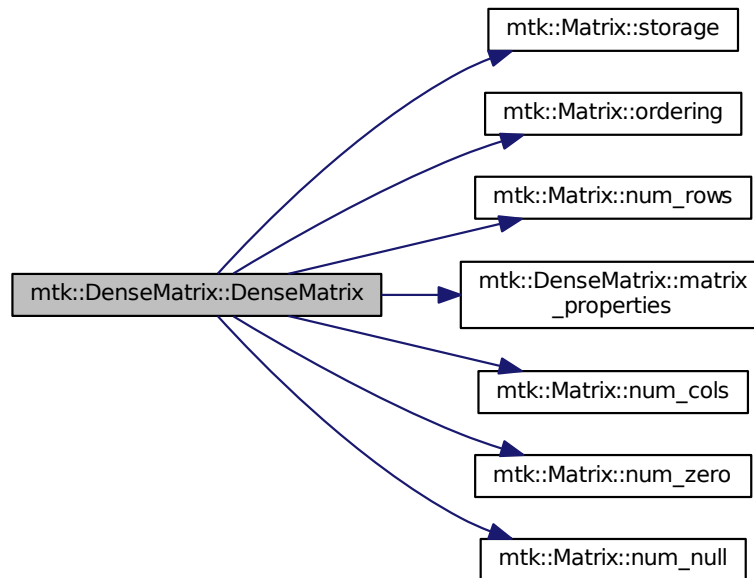
#### 17.3.2.2 `mtk::DenseMatrix::DenseMatrix ( const DenseMatrix &in )`

##### Parameters

<code>in</code>	<i>in</i>	Given matrix.
-----------------	-----------	---------------

Definition at line 173 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



#### 17.3.2.3 mtk::DenseMatrix::DenseMatrix ( const int & *num\_rows*, const int & *num\_cols* )

##### Parameters

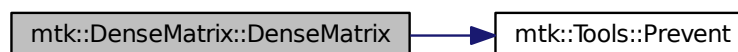
in	<i>num_rows</i>	Number of rows of the required matrix.
in	<i>num_cols</i>	Number of rows of the required matrix.

##### Exceptions

<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 206 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



### 17.3.2.4 mtk::DenseMatrix::DenseMatrix ( const int & *rank*, const bool & *padded*, const bool & *transpose* )

Used in the construction of the mimetic operators.

Def\*\*. A **padded matrix** is a matrix with its first and last rows initialized to only zero values:

$$\bar{\mathbf{I}} = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

#### Parameters

in	<i>rank</i>	Rank or number of rows/cols in square matrix.
in	<i>padded</i>	Should it be padded?
in	<i>transpose</i>	Should I return the transpose of the requested matrix?

#### Exceptions

<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 228 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



### 17.3.2.5 mtk::DenseMatrix::DenseMatrix ( const Real \*const *gen*, const int & *gen\_length*, const int & *pro\_length*, const bool & *transpose* )

Def\*\*. In linear algebra, a **Vandermonde matrix** is a matrix with terms of a geometric progression in each row. This progression uses the terms of a given **generator vector**:

$$\mathbf{V} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \dots & \alpha_m^{n-1} \end{pmatrix}$$

This constructor generates a Vandermonde matrix, as defined above.

Obs\*\*. It is important to understand that the generator vectors to be used are nothing but a very particular instance of a grid. These are little chunks, little samples, if you will, of a grid which is rectangular and uniform. So the selected samples, on the [mtk::Div1D](#) and [mtk::Grad1D](#), basically represent the entire space, the entire grid. This is why neither the CRS nor the CBS algorithms may work for irregular geometries, such as curvilinear grids.

## Parameters

in	<i>gen</i>	Given generator vector.
in	<i>gen_length</i>	Length generator vector.
in	<i>pro_length</i>	Length the progression.
in	<i>transpose</i>	Should the transpose be created instead?

## Exceptions

<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 269 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



## 17.3.2.6 mtk::DenseMatrix::~~DenseMatrix ( )

Definition at line 317 of file [mtk\\_dense\\_matrix.cc](#).

## 17.3.3 Member Function Documentation

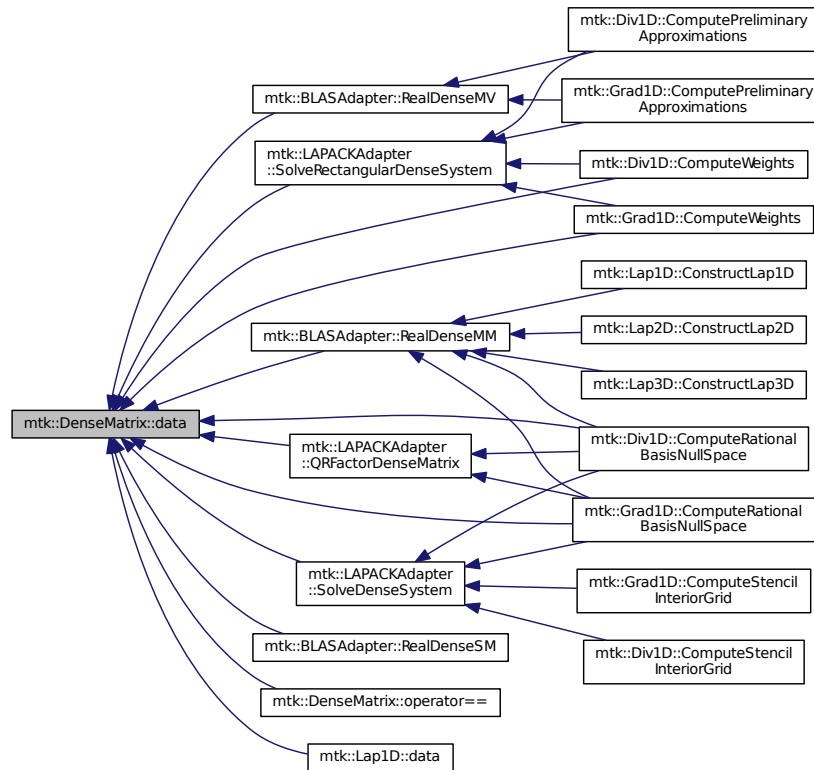
## 17.3.3.1 mtk::Real \* mtk::DenseMatrix::data ( ) const [noexcept]

## Returns

Pointer to an array of [mtk::Real](#).

Definition at line 349 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



### 17.3.3.2 `mtk::Real mtk::DenseMatrix::GetValue ( const int & row_coord, const int & col_coord ) const` [noexcept]

## Parameters

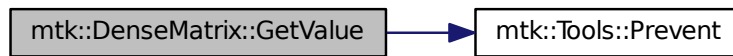
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.

**Returns**

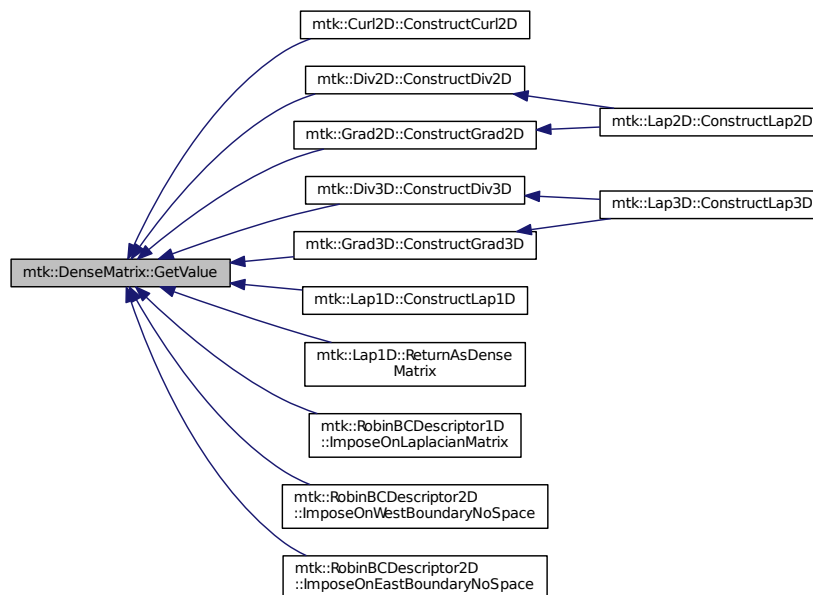
The required value at the specified coordinates.

Definition at line 354 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.3.3.3 `mtk::DenseMatrix mtk::DenseMatrix::Kron ( const DenseMatrix & aa, const DenseMatrix & bb ) [static]`

**Parameters**

<code>in</code>	<code>aa</code>	First matrix.
-----------------	-----------------	---------------

<code>in</code>	<code>bb</code>	Second matrix.
-----------------	-----------------	----------------

#### Exceptions

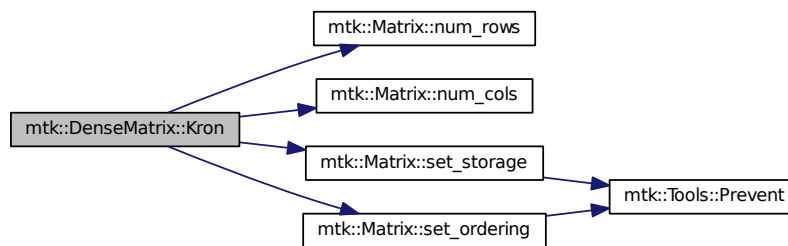
<code>std::bad_alloc</code>
-----------------------------

**Todo** Implement Kronecker product using the BLAS.

**Todo** Implement Kron using the BLAS.

Definition at line 496 of file `mtk_dense_matrix.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



**17.3.3.4** `mtk::Matrix mtk::DenseMatrix::matrix_properties ( ) const [noexcept]`

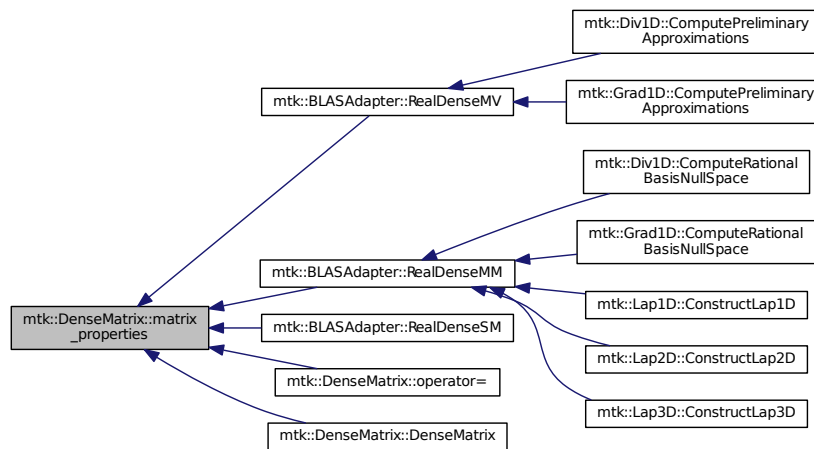


## Returns

Pointer to a [Matrix](#).

Definition at line 323 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



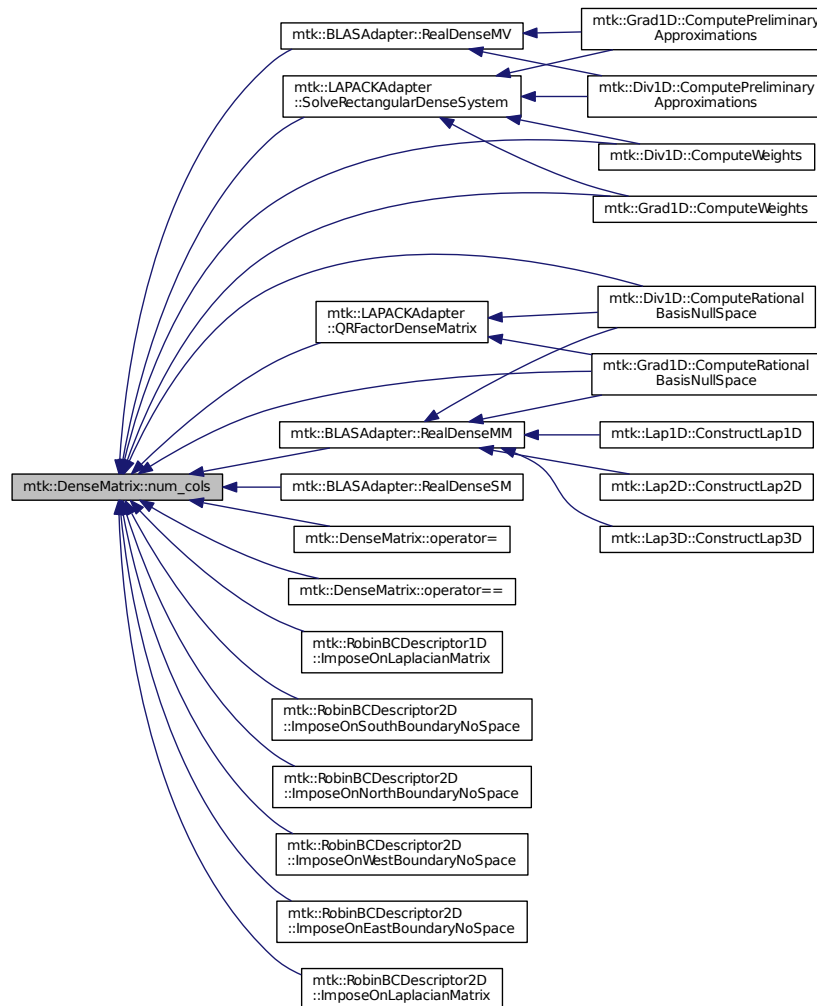
### 17.3.3.5 int mtk::DenseMatrix::num\_cols ( ) const [noexcept]

## Returns

Number of columns of the matrix.

Definition at line 344 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



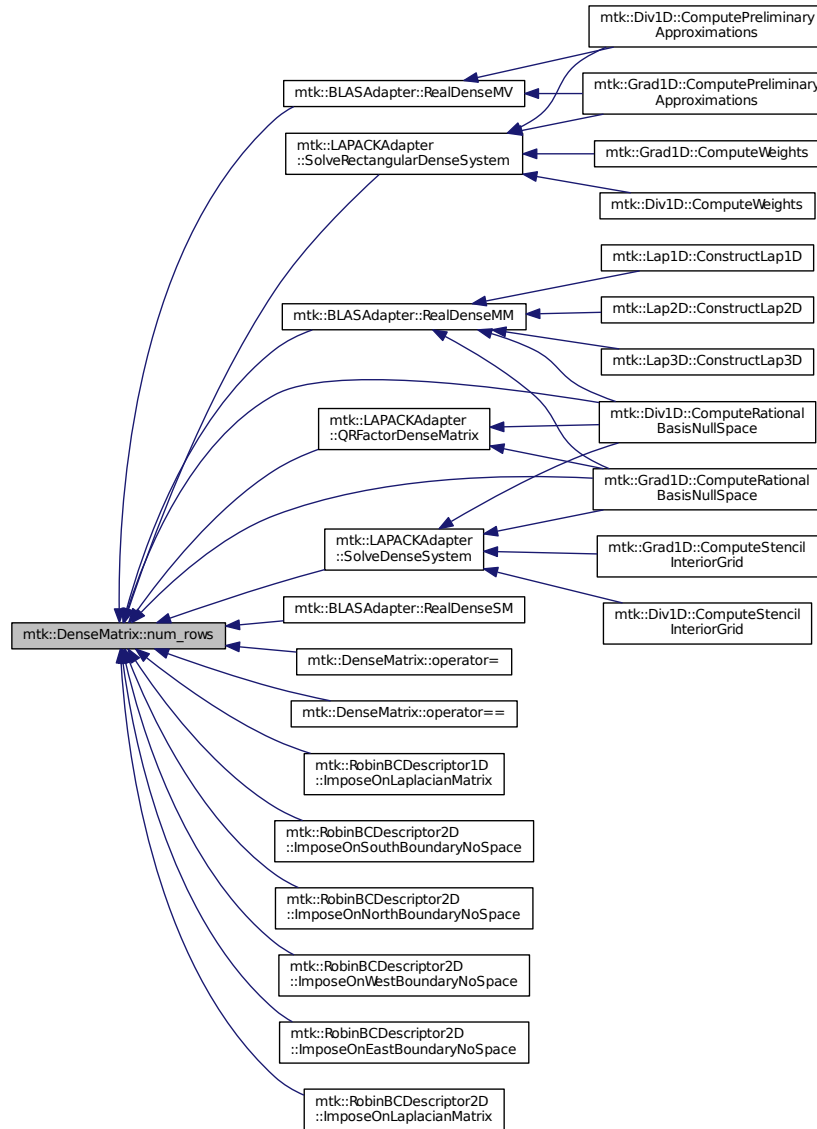
17.3.3.6 `int mtk::DenseMatrix::num_rows ( ) const [noexcept]`

## Returns

Number of rows of the matrix.

Definition at line 339 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



### 17.3.3.7 mtk::DenseMatrix & mtk::DenseMatrix::operator= ( const DenseMatrix & in )

## Parameters

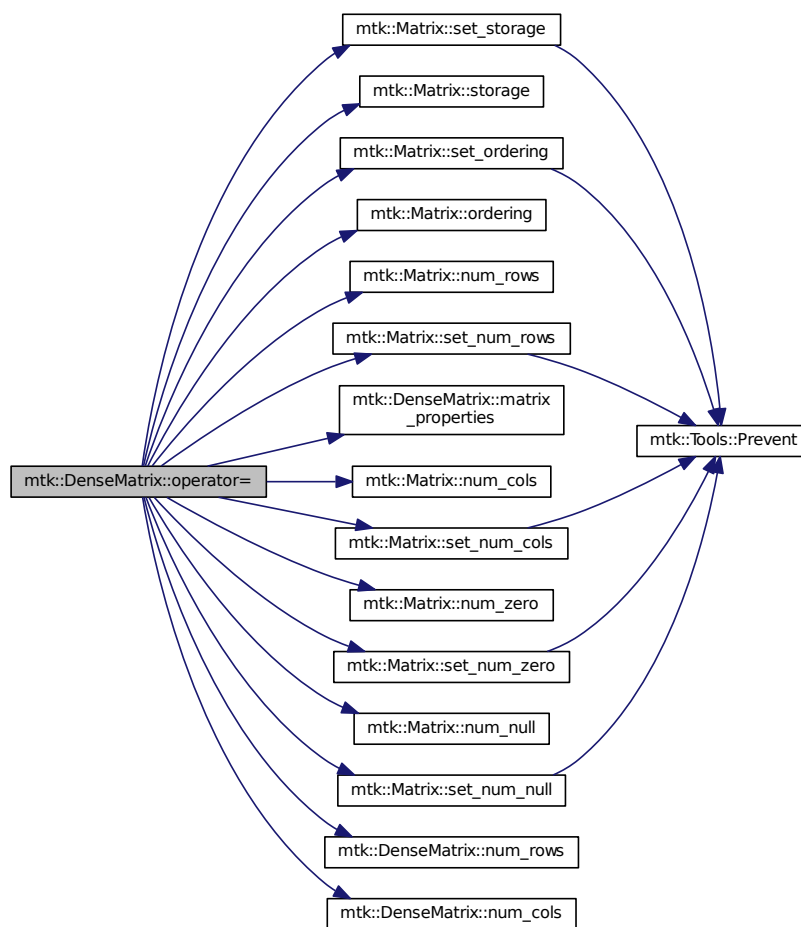
<i>in</i>	<i>in</i>	Given matrix.
-----------	-----------	---------------

## Returns

Copy of the given matrix.

Definition at line 105 of file [mtk\\_dense\\_matrix.cc](#).

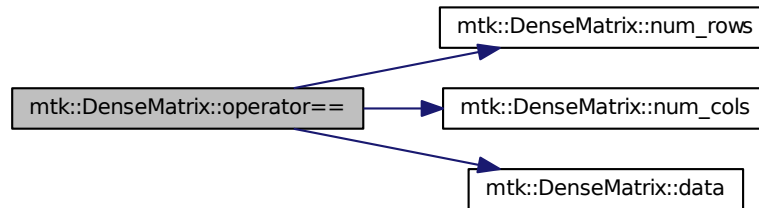
Here is the call graph for this function:



17.3.3.8 `bool mtk::DenseMatrix::operator==( const DenseMatrix & in )`

Definition at line 146 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:

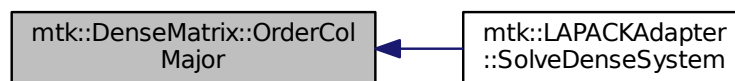


17.3.3.9 `void mtk::DenseMatrix::OrderColMajor ( )`

**Todo** Improve this so that no new arrays have to be created.

Definition at line 457 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:

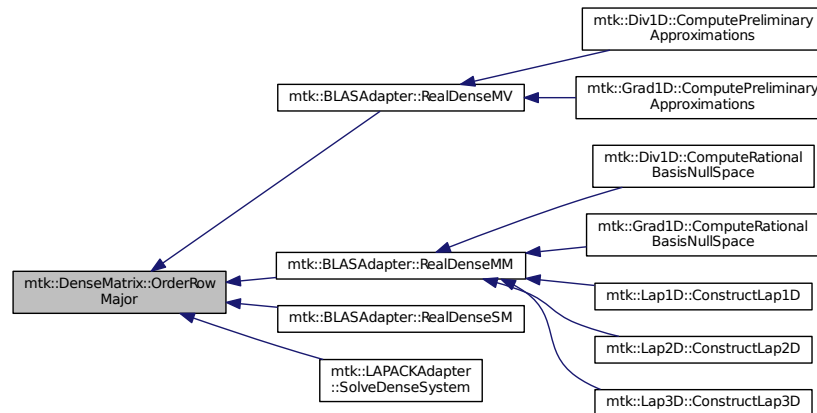


17.3.3.10 `void mtk::DenseMatrix::OrderRowMajor ( )`

**Todo** Improve this so that no new arrays have to be created.

Definition at line 416 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



17.3.3.11 void mtk::DenseMatrix::SetOrdering ( mtk::MatrixOrdering oo ) [noexcept]

#### Parameters

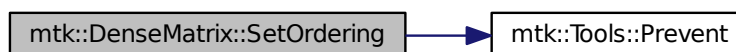
in	oo	Ordering.
----	----	-----------

#### Returns

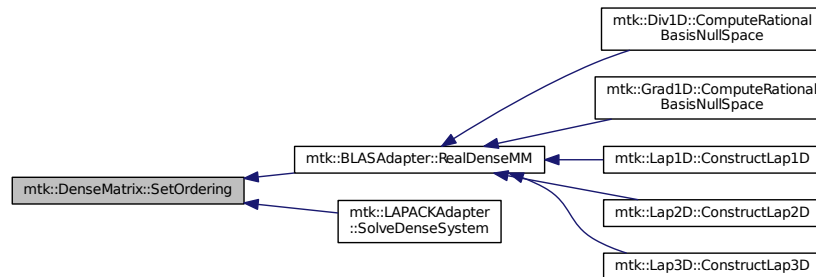
The required value at the specified coordinates.

Definition at line 328 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



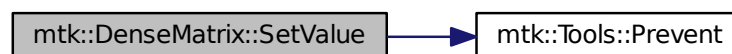
17.3.3.12 `void mtk::DenseMatrix::SetValue ( const int & row_coord, const int & col_coord, const Real & val )` [noexcept]

#### Parameters

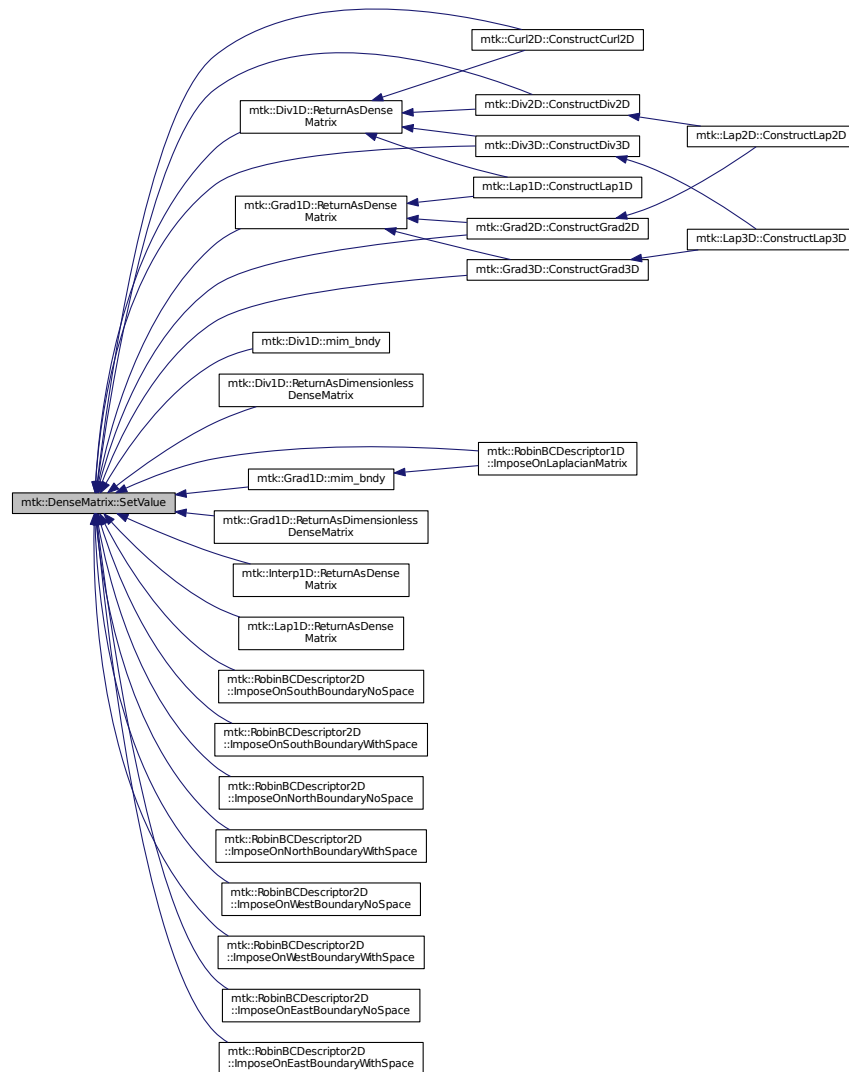
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.
in	<i>val</i>	Row Actual value to be inserted.

Definition at line 366 of file `mtk_dense_matrix.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



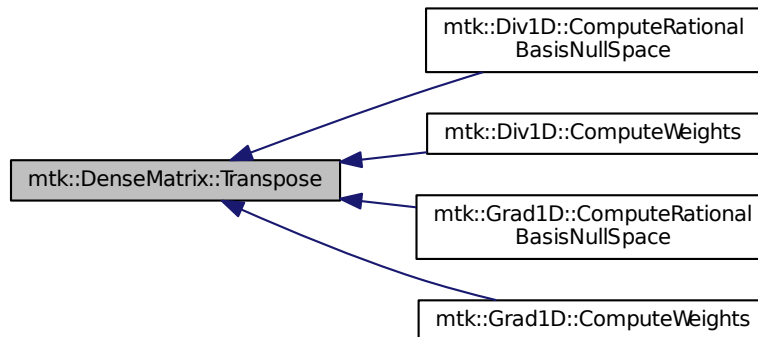
17.3.3.13 void mtk::DenseMatrix::Transpose ( )

**Todo** Improve this so that no extra arrays have to be created.

Definition at line 379 of file [mtk\\_dense\\_matrix.cc](#).



Here is the caller graph for this function:



17.3.3.14 `bool mtk::DenseMatrix::WriteToFile ( const std::string & filename ) const`

#### Parameters

<code>in</code>	<code>filename</code>	Name of the output file.
-----------------	-----------------------	--------------------------

#### Returns

Success of the file writing process.

#### See also

<http://www.gnuplot.info/>

Definition at line 539 of file `mtk_dense_matrix.cc`.

### 17.3.4 Friends And Related Function Documentation

17.3.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::DenseMatrix & in )` `[friend]`

Definition at line 79 of file `mtk_dense_matrix.cc`.

### 17.3.5 Member Data Documentation

17.3.5.1 `Real* mtk::DenseMatrix::data_` `[private]`

Definition at line 291 of file `mtk_dense_matrix.h`.

### 17.3.5.2 Matrix `mtk::DenseMatrix::matrix_properties_` [private]

Definition at line 289 of file [mtk\\_dense\\_matrix.h](#).

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

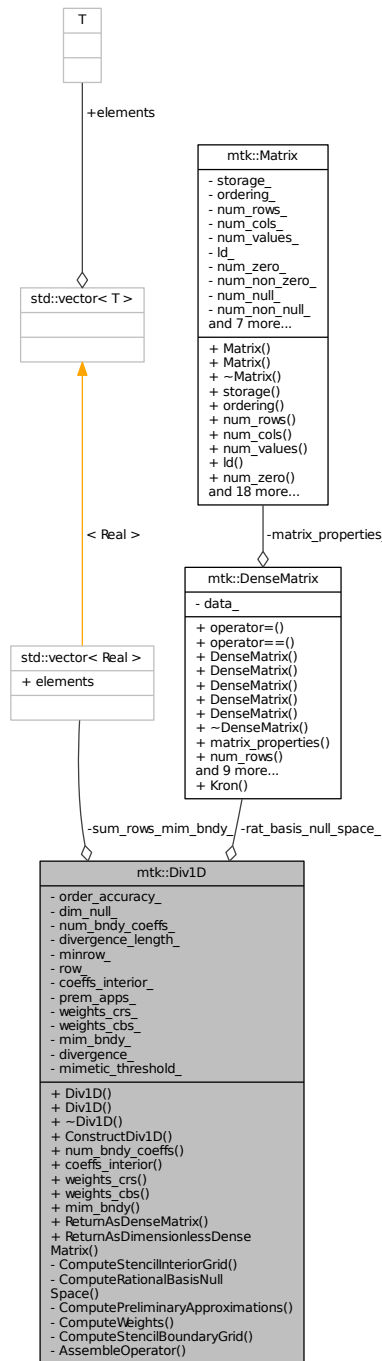
- [include/mtk\\_dense\\_matrix.h](#)
- [src/mtk\\_dense\\_matrix.cc](#)

## 17.4 `mtk::Div1D` Class Reference

Implements a 1D mimetic divergence operator.

```
#include <mtk_div_1d.h>
```

Collaboration diagram for mtk::Div1D:



## Public Member Functions

- [Div1D](#) ()

*Default constructor.*

- `Div1D` (const `Div1D` &div)

*Copy constructor.*

- `~Div1D` ()

*Destructor.*

- bool `ConstructDiv1D` (int order\_accuracy=`kDefaultOrderAccuracy`, `Real` mimetic\_threshold=`kDefaultMimeticThreshold`)

*Factory method implementing the CBS Algorithm to build operator.*

- int `num_bndy_coeffs` () const

*Returns how many coefficients are approximating at the boundary.*

- `Real` \* `coeffs_interior` () const

*Returns coefficients for the interior of the grid.*

- `Real` \* `weights_crs` (void) const

*Return collection of weights as computed by the CRSA.*

- `Real` \* `weights_cbs` (void) const

*Return collection of weights as computed by the CBSA.*

- `DenseMatrix` `mim_bndy` () const

*Return collection of mimetic approximations at the boundary.*

- `DenseMatrix` `ReturnAsDenseMatrix` (const `UniStgGrid1D` &grid) const

*Return the operator as a dense matrix.*

- `DenseMatrix` `ReturnAsDimensionlessDenseMatrix` (int num\_cells\_x) const

*Returns the operator as a dimensionless dense matrix.*

## Private Member Functions

- bool `ComputeStencilInteriorGrid` (void)

*Stage 1 of the CBS Algorithm.*

- bool `ComputeRationalBasisNullSpace` (void)

*Stage 2.1 of the CBS Algorithm.*

- bool `ComputePreliminaryApproximations` (void)

*Stage 2.2 of the CBS Algorithm.*

- bool `ComputeWeights` (void)

*Stage 2.3 of the CBS Algorithm.*

- bool `ComputeStencilBoundaryGrid` (void)

*Stage 2.4 of the CBS Algorithm.*

- bool `AssembleOperator` (void)

*Stage 3 of the CBS Algorithm.*

## Private Attributes

- int `order_accuracy_`

*Order of numerical accuracy of the operator.*

- int `dim_null_`

*Dim. null-space for boundary approximations.*

- int `num_bndy_coeffs_`

*Req. coeffs. per bndy pt. uni. order accuracy.*

- int `divergence_length_`

- Length of the output array.*
- int [minrow\\_](#)  
*Row from the optimizer with the minimum rel. nor.*
- int [row\\_](#)  
*Row currently processed by the optimizer.*
- [DenseMatrix](#) [rat\\_basis\\_null\\_space\\_](#)  
*Rational b. null-space w. bndy.*
- [Real](#) \* [coeffs\\_interior\\_](#)  
*Interior stencil.*
- [Real](#) \* [prem\\_apps\\_](#)  
*2D array of boundary preliminary approximations.*
- [Real](#) \* [weights\\_crs\\_](#)  
*Array containing weights from CRSA.*
- [Real](#) \* [weights\\_cbs\\_](#)  
*Array containing weights from CBSA.*
- [Real](#) \* [mim\\_bndy\\_](#)  
*Array containing mimetic boundary approximations.*
- [Real](#) \* [divergence\\_](#)  
*Output array containing the operator and weights.*
- [std::vector](#)< [Real](#) > [sum\\_rows\\_mim\\_bndy\\_](#)  
*Sum of the boundary rows.*
- [Real](#) [mimetic\\_threshold\\_](#)  
*< Mimetic threshold.*

## Friends

- [std::ostream](#) & [operator<<](#) ([std::ostream](#) &stream, [Div1D](#) &in)  
*Output stream operator for printing.*

### 17.4.1 Detailed Description

This class implements a 1D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 83 of file [mtk\\_div\\_1d.h](#).

### 17.4.2 Constructor & Destructor Documentation

#### 17.4.2.1 [mtk::Div1D::Div1D \( \)](#)

Definition at line 137 of file [mtk\\_div\\_1d.cc](#).

#### 17.4.2.2 [mtk::Div1D::Div1D \( const \[Div1D\]\(#\) &div \)](#)

## Parameters

<i>in</i>	<i>div</i>	Given divergence.
-----------	------------	-------------------

Definition at line 152 of file [mtk\\_div\\_1d.cc](#).

#### 17.4.2.3 mtk::Div1D::~~Div1D ( )

Definition at line 167 of file [mtk\\_div\\_1d.cc](#).

### 17.4.3 Member Function Documentation

#### 17.4.3.1 bool mtk::Div1D::AssembleOperator ( void ) [private]

Construct the output array with the operator and its weights.

1. The first entry of the array will contain the order of accuracy.
2. The second entry the collection of coefficients for interior of grid.
3. If `order_accuracy_ > 2`, then third entry is the collection of weights.
4. If `order_accuracy_ > 2`, next `dim_null_` entries is approximating coefficients for the west boundary of the grid.

Definition at line 1459 of file [mtk\\_div\\_1d.cc](#).

#### 17.4.3.2 mtk::Real \* mtk::Div1D::coeffs\_interior ( ) const

##### Returns

Coefficients for the interior of the grid.

Definition at line 332 of file [mtk\\_div\\_1d.cc](#).

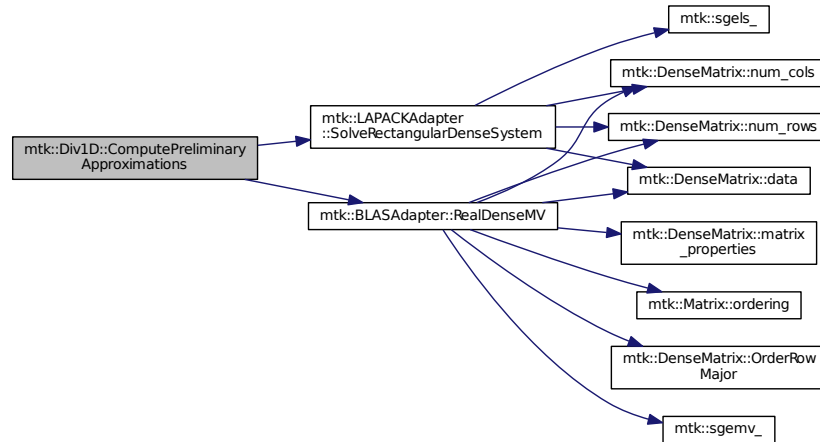
#### 17.4.3.3 bool mtk::Div1D::ComputePreliminaryApproximations ( void ) [private]

Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.
2. Compute the `dim_null` near-the-boundary columns of the pi matrix.
3. Create the Vandermonde matrix for this iteration.
4. New order-selector vector (gets re-written with LAPACK solutions).
5. Solving  $TT*rr = ob$  yields the columns `rr` of the KK matrix.
6. Scale the KK matrix to make it a rational basis for null-space.
7. Extract the last `dim_null` values of the pre-scaled `ob`.
8. Once we posses the bottom elements, we proceed with the scaling.

Definition at line 765 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



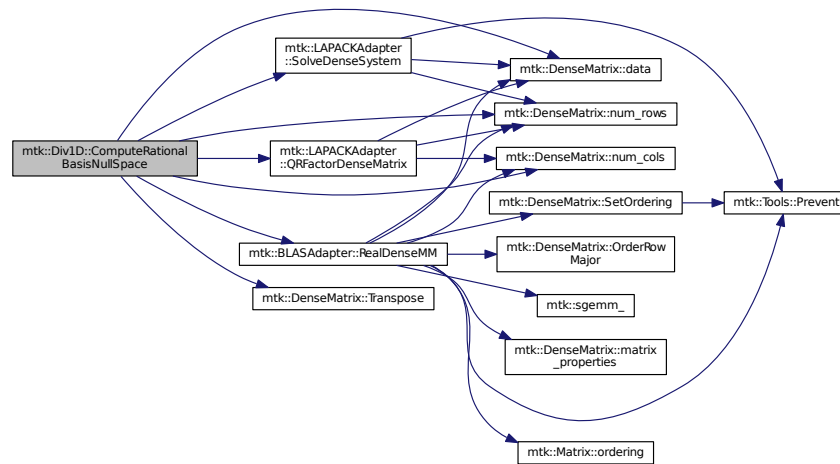
#### 17.4.3.4 `bool mtk::Div1D::ComputeRationalBasisNullSpace ( void ) [private]`

Compute a rational basis for the null-space of the Vandermonde matrix approximating at the west boundary.

1. Create generator vector for the first approximation.
2. Create Vandermonde matrix.
3. QR-factorize the Vandermonde matrix.
4. Extract the basis for the null-space from Q matrix.
5. Scale null-space to make it rational.

Definition at line 589 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



#### 17.4.3.5 `bool mtk::Div1D::ComputeStencilBoundaryGrid ( void ) [private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.
2. Compute alpha values.
3. Compute the mimetic boundary approximations.

Definition at line 1358 of file [mtk\\_div\\_1d.cc](#).

#### 17.4.3.6 `bool mtk::Div1D::ComputeStencilInteriorGrid ( void ) [private]`

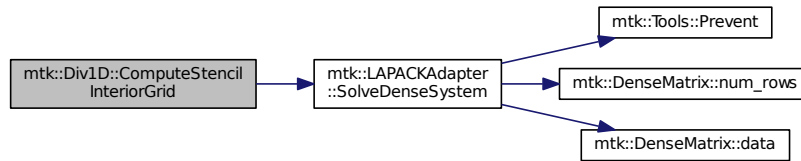
Compute the stencil approximating the interior of the staggered grid.

1. Create vector for interior spatial coordinates.
2. Create Vandermonde matrix (using interior coordinates as generator).
3. Create order-selector vector.
4. Solve dense Vandermonde system to attain the interior coefficients.

Definition at line 488 of file [mtk\\_div\\_1d.cc](#).



Here is the call graph for this function:



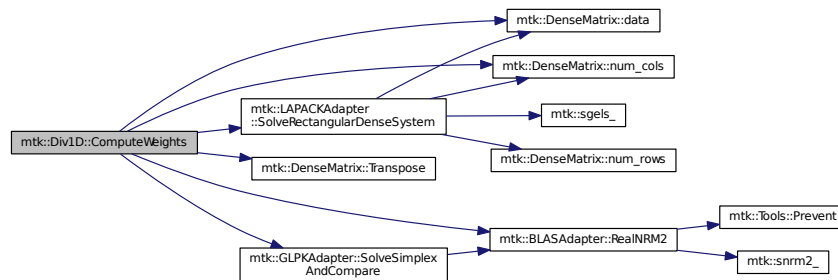
#### 17.4.3.7 bool mtk::Div1D::ComputeWeights ( void ) [private]

Compute the set of mimetic weights to impose the mimetic condition.

1. Construct the  $\mathbf{A}$  matrix.
2. Use interior stencil to build proper RHS vector  $\mathbf{h}$ .
3. Get weights (as **CRSA**):  $\mathbf{A}\mathbf{q} = \mathbf{h}$ .
4. If required order is greater than critical order, start the **CBSA**.
5. Create  $\mathbf{B}$  matrix from  $\mathbf{A}$ .
6. Prepare constraint vector as in the CBSA:  $\mathbf{c}$ .
7. Brute force search through all the rows of the  $\Phi$  matrix.
8. Apply solution found from brute force search.

Definition at line 985 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



17.4.3.8 `bool mtk::Div1D::ConstructDiv1D ( int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

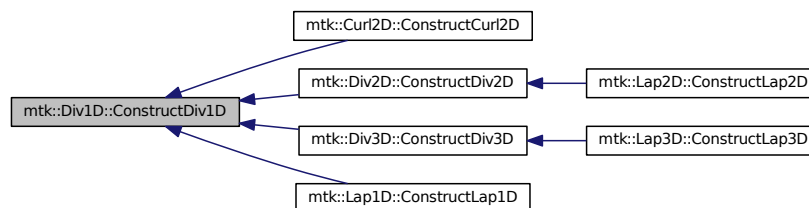
1. Compute stencil for the interior cells.
2. Compute a rational basis for the null-space for the first matrix.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.
6. Assemble operator.

Definition at line 188 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



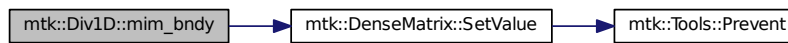
17.4.3.9 `mtk::DenseMatrix mtk::Div1D::mim_bndy ( ) const`

**Returns**

Collection of mimetic approximations at the boundary.

Definition at line 347 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:

**17.4.3.10 int mtk::Div1D::num\_bndy\_coeffs ( ) const****Returns**

How many coefficients are approximating at the boundary.

Definition at line 327 of file [mtk\\_div\\_1d.cc](#).

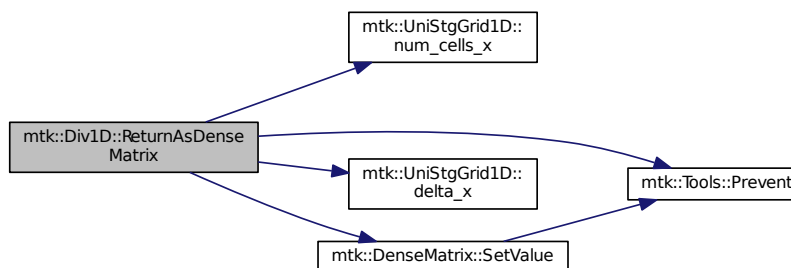
**17.4.3.11 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix ( const UniStgGrid1D & grid ) const****Returns**

The operator as a dense matrix.

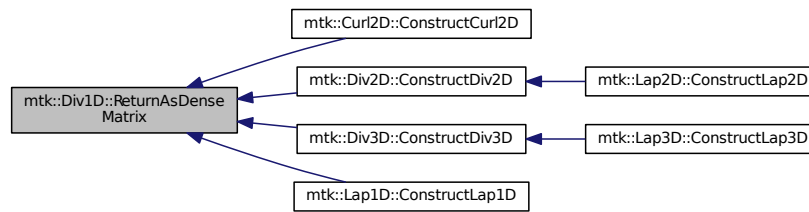
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 362 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 17.4.3.12 `mtk::DenseMatrix mtk::Div1D::ReturnAsDimensionlessDenseMatrix ( int num_cells_x ) const`

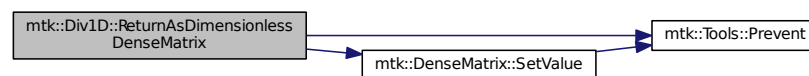
##### Returns

The operator as a dimensionless dense matrix.

1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 426 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



#### 17.4.3.13 `mtk::Real * mtk::Div1D::weights_cbs ( void ) const`

##### Returns

Collection of weights as computed by the CBSA.

Definition at line 342 of file [mtk\\_div\\_1d.cc](#).

#### 17.4.3.14 `mtk::Real * mtk::Div1D::weights_crs ( void ) const`

##### Returns

Collection of weights as computed by the CRSA.

Definition at line 337 of file [mtk\\_div\\_1d.cc](#).

### 17.4.4 Friends And Related Function Documentation

17.4.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::Div1D & in )` [*friend*]

1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

Definition at line 84 of file [mtk\\_div\\_1d.cc](#).

### 17.4.5 Member Data Documentation

17.4.5.1 `Real* mtk::Div1D::coeffs_interior_` [*private*]

Definition at line 211 of file [mtk\\_div\\_1d.h](#).

17.4.5.2 `int mtk::Div1D::dim_null_` [*private*]

Definition at line 203 of file [mtk\\_div\\_1d.h](#).

17.4.5.3 `Real* mtk::Div1D::divergence_` [*private*]

Definition at line 216 of file [mtk\\_div\\_1d.h](#).

17.4.5.4 `int mtk::Div1D::divergence_length_` [*private*]

Definition at line 205 of file [mtk\\_div\\_1d.h](#).

17.4.5.5 `Real* mtk::Div1D::mim_bndy_` [*private*]

Definition at line 215 of file [mtk\\_div\\_1d.h](#).

17.4.5.6 `Real mtk::Div1D::mimetic_threshold_` [*private*]

Definition at line 220 of file [mtk\\_div\\_1d.h](#).

17.4.5.7 `int mtk::Div1D::minrow_` [*private*]

Definition at line 206 of file [mtk\\_div\\_1d.h](#).

17.4.5.8 `int mtk::Div1D::num_bndy_coeffs_` [*private*]

Definition at line 204 of file [mtk\\_div\\_1d.h](#).

17.4.5.9 `int mtk::Div1D::order_accuracy_ [private]`

Definition at line 202 of file [mtk\\_div\\_1d.h](#).

17.4.5.10 `Real* mtk::Div1D::prem_apps_ [private]`

Definition at line 212 of file [mtk\\_div\\_1d.h](#).

17.4.5.11 `DenseMatrix mtk::Div1D::rat_basis_null_space_ [private]`

Definition at line 209 of file [mtk\\_div\\_1d.h](#).

17.4.5.12 `int mtk::Div1D::row_ [private]`

Definition at line 207 of file [mtk\\_div\\_1d.h](#).

17.4.5.13 `std::vector<Real> mtk::Div1D::sum_rows_mim_bndy_ [private]`

Definition at line 218 of file [mtk\\_div\\_1d.h](#).

17.4.5.14 `Real* mtk::Div1D::weights_cbs_ [private]`

Definition at line 214 of file [mtk\\_div\\_1d.h](#).

17.4.5.15 `Real* mtk::Div1D::weights_crs_ [private]`

Definition at line 213 of file [mtk\\_div\\_1d.h](#).

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

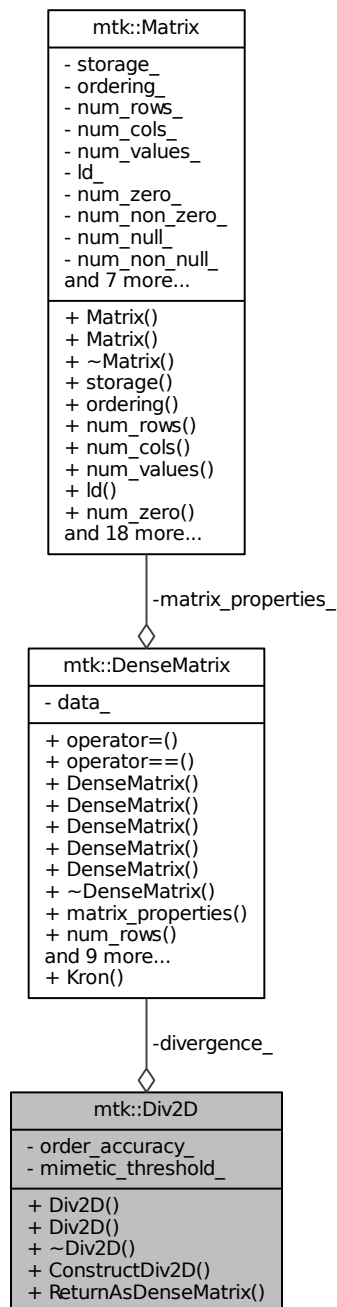
- [include/mtk\\_div\\_1d.h](#)
- [src/mtk\\_div\\_1d.cc](#)

## 17.5 mtk::Div2D Class Reference

Implements a 2D mimetic divergence operator.

```
#include <mtk_div_2d.h>
```

Collaboration diagram for mtk::Div2D:



## Public Member Functions

- [Div2D \(\)](#)

*Default constructor.*

- [Div2D](#) (const [Div2D](#) &div)

*Copy constructor.*

- [~Div2D](#) ()

*Destructor.*

- bool [ConstructDiv2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_↔ threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix divergence\\_](#)

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.5.1 Detailed Description

This class implements a 2D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 76 of file [mtk\\_div\\_2d.h](#).

## 17.5.2 Constructor & Destructor Documentation

### 17.5.2.1 [mtk::Div2D::Div2D](#) ( )

Definition at line 69 of file [mtk\\_div\\_2d.cc](#).

### 17.5.2.2 [mtk::Div2D::Div2D](#) ( const [Div2D](#) &div )

#### Parameters

<a href="#">in</a>	<a href="#">div</a>	Given divergence.
--------------------	---------------------	-------------------

Definition at line 73 of file [mtk\\_div\\_2d.cc](#).

### 17.5.2.3 [mtk::Div2D::~~Div2D](#) ( )

Definition at line 77 of file [mtk\\_div\\_2d.cc](#).



### 17.5.3 Member Function Documentation

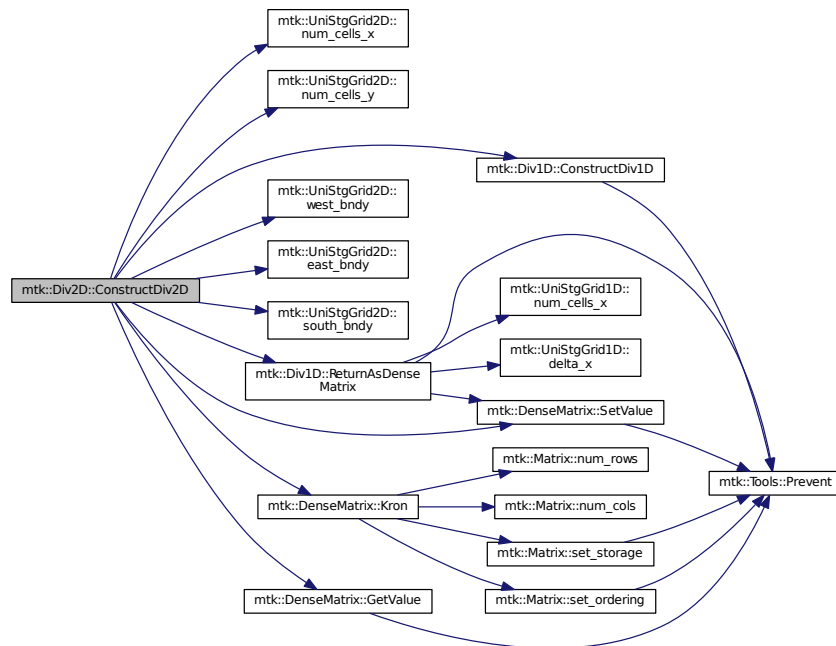
17.5.3.1 `bool mtk::Div2D::ConstructDiv2D ( const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

Definition at line 79 of file [mtk\\_div\\_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.5.3.2 `mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix ( ) const`

**Returns**

The operator as a dense matrix.

Definition at line 147 of file [mtk\\_div\\_2d.cc](#).

Here is the caller graph for this function:

**17.5.4 Member Data Documentation****17.5.4.1 DenseMatrix mtk::Div2D::divergence\_ [private]**

Definition at line 108 of file [mtk\\_div\\_2d.h](#).

**17.5.4.2 Real mtk::Div2D::mimetic\_threshold\_ [private]**

Definition at line 112 of file [mtk\\_div\\_2d.h](#).

**17.5.4.3 int mtk::Div2D::order\_accuracy\_ [private]**

Definition at line 110 of file [mtk\\_div\\_2d.h](#).

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

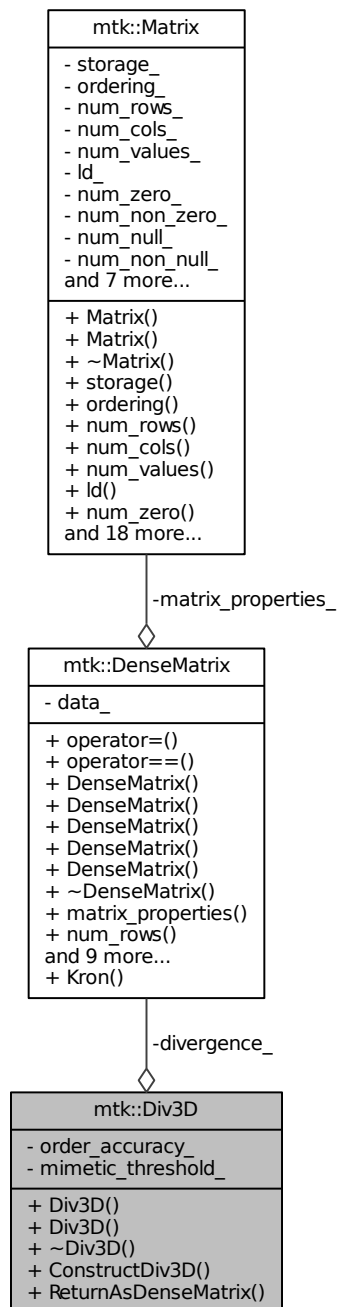
- [include/mtk\\_div\\_2d.h](#)
- [src/mtk\\_div\\_2d.cc](#)

**17.6 mtk::Div3D Class Reference**

Implements a 3D mimetic divergence operator.

```
#include <mtk_div_3d.h>
```

Collaboration diagram for mtk::Div3D:



## Public Member Functions

- [Div3D\(\)](#)

*Default constructor.*

- [Div3D](#) (const [Div3D](#) &div)

*Copy constructor.*

- [~Div3D](#) ()

*Destructor.*

- bool [ConstructDiv3D](#) (const [UniStgGrid3D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_ $\leftrightarrow$  threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix](#) divergence\_

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real](#) [mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.6.1 Detailed Description

This class implements a 3D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 76 of file [mtk\\_div\\_3d.h](#).

## 17.6.2 Constructor & Destructor Documentation

### 17.6.2.1 [mtk::Div3D::Div3D](#) ( )

Definition at line 67 of file [mtk\\_div\\_3d.cc](#).

### 17.6.2.2 [mtk::Div3D::Div3D](#) ( const [Div3D](#) &div )

#### Parameters

<a href="#">in</a>	<a href="#">div</a>	Given divergence.
--------------------	---------------------	-------------------

Definition at line 71 of file [mtk\\_div\\_3d.cc](#).

### 17.6.2.3 [mtk::Div3D::~~Div3D](#) ( )

Definition at line 75 of file [mtk\\_div\\_3d.cc](#).

### 17.6.3 Member Function Documentation

17.6.3.1 `bool mtk::Div3D::ConstructDiv3D ( const UniStgGrid3D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

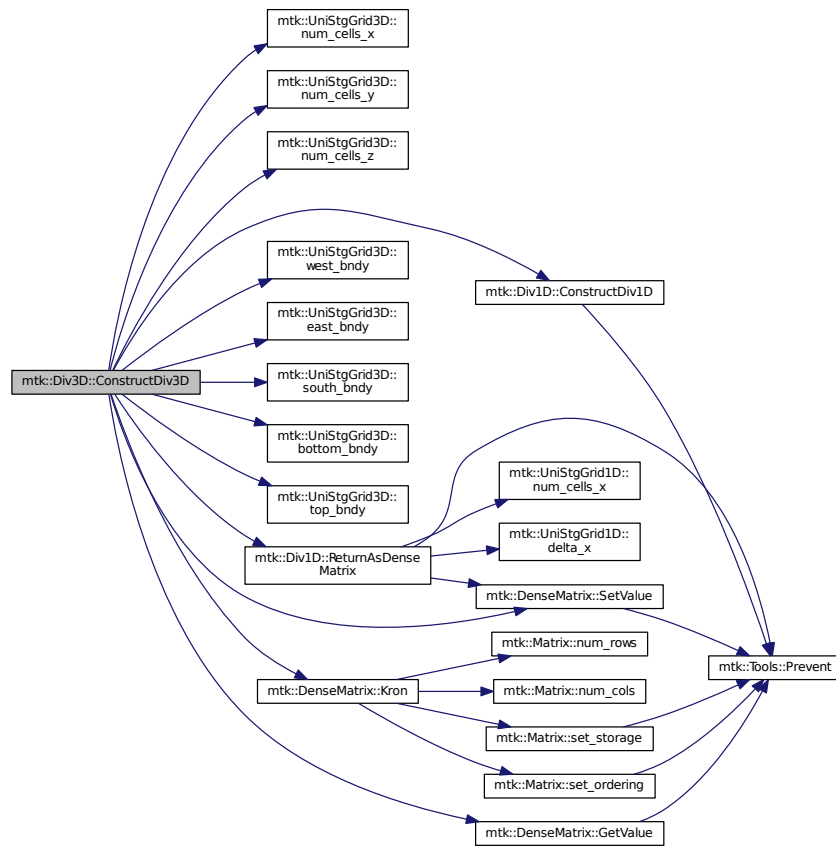
Returns

Success of the construction.

1. Build preliminary staggering through the x direction.
2. Build preliminary staggering through the y direction.
3. Build preliminary staggering through the z direction.
4. Actual operator:  $DD_{xyz} = [dx \ dy \ dz]$ .

Definition at line 77 of file [mtk\\_div\\_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 17.6.3.2 `mtk::DenseMatrix mtk::Div3D::ReturnAsDenseMatrix ( ) const`

##### Returns

The operator as a dense matrix.

Definition at line 186 of file [mtk\\_div\\_3d.cc](#).

Here is the caller graph for this function:



### 17.6.4 Member Data Documentation

#### 17.6.4.1 `DenseMatrix mtk::Div3D::divergence_ [private]`

Definition at line 108 of file [mtk\\_div\\_3d.h](#).

#### 17.6.4.2 `Real mtk::Div3D::mimetic_threshold_ [private]`

Definition at line 112 of file [mtk\\_div\\_3d.h](#).

#### 17.6.4.3 `int mtk::Div3D::order_accuracy_ [private]`

Definition at line 110 of file [mtk\\_div\\_3d.h](#).

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

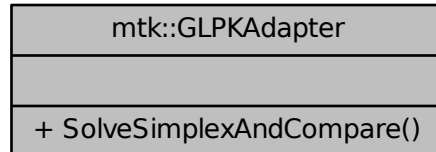
- [include/mtk\\_div\\_3d.h](#)
- [src/mtk\\_div\\_3d.cc](#)

## 17.7 mtk::GLPKAdapter Class Reference

Adapter class for the GLPK API.

```
#include <mtk_glpk_adapter.h>
```

Collaboration diagram for mtk::GLPKAdapter:



### Static Public Member Functions

- static `mtk::Real SolveSimplexAndCompare (mtk::Real *A, int nrows, int ncols, int kk, mtk::Real *hh, mtk::Real *qq, int robjective, mtk::Real mimetic_tol, int copy)`

*Solves a CLO problem and compares the solution to a reference solution.*

#### 17.7.1 Detailed Description

This class contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the GLPK.

The **GLPK (GNU Linear Programming Kit)** package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

#### Warning

We use the GLPK temporarily in order to test the CBSA, but it will be removed due to potential licensing issues.

#### See also

<http://www.gnu.org/software/glpk/>

**Todo** Rescind from the GLPK as the numerical core for CLO problems.

Definition at line 102 of file `mtk_glpk_adapter.h`.

#### 17.7.2 Member Function Documentation

17.7.2.1 `mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare ( mtk::Real * A, int nrows, int ncols, int kk, mtk::Real * hh, mtk::Real * qq, int robjective, mtk::Real mimetic_tol, int copy ) [static]`

This routine is the pivot of the CBSA. It solves a Constrained Linear Optimization (CLO) problem, and it compares the attained solution to a given reference solution. This comparison is done computing the norm-2 relative error.



## Parameters

in	<i>alpha</i>	First scalar.
in	<i>AA</i>	Given matrix.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in	<i>beta</i>	Second scalar.

## Returns

Relative error computed between attained solution and provided ref.

## Warning

GLPK indexes in [1,n], so we must get the extra space needed.

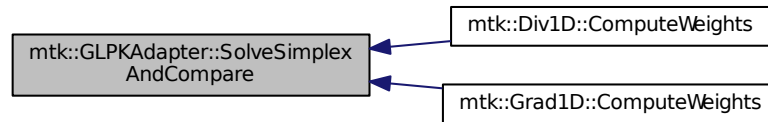
1. Memory allocation.
2. Fill the problem.
3. Copy the row to the vector objective.
4. Forming the RHS.
5. Setting up the objective function.
6. Setting up constraints.
7. Copy the matrix minus the row objective to the glpk problem.
8. Solve problem.

Definition at line 77 of file [mtk\\_glpk\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

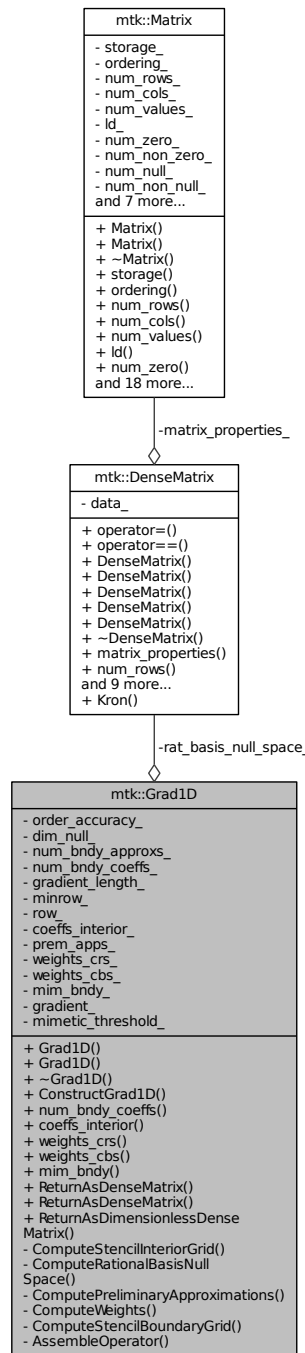
- [include/mtk\\_glpk\\_adapter.h](#)
- [src/mtk\\_glpk\\_adapter.cc](#)

## 17.8 mtk::Grad1D Class Reference

Implements a 1D mimetic gradient operator.

```
#include <mtk_grad_1d.h>
```

Collaboration diagram for mtk::Grad1D:



## Public Member Functions

- [Grad1D](#) ()

- *Default constructor.*
- [Grad1D](#) (const [Grad1D](#) &grad)
- *Copy constructor.*
- [~Grad1D](#) ()
- *Destructor.*
- bool [ConstructGrad1D](#) (int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_threshold=[kDefaultMimeticThreshold](#))
- *Factory method implementing the CBS Algorithm to build operator.*
- int [num\\_bndy\\_coeffs](#) () const
- *Returns how many coefficients are approximating at the boundary.*
- [Real](#) \* [coeffs\\_interior](#) () const
- *Returns coefficients for the interior of the grid.*
- [Real](#) \* [weights\\_crs](#) (void) const
- *Returns collection of weights as computed by the CRSA.*
- [Real](#) \* [weights\\_cbs](#) (void) const
- *Returns collection of weights as computed by the CBSA.*
- [DenseMatrix](#) [mim\\_bndy](#) () const
- *Return collection of mimetic approximations at the boundary.*
- [DenseMatrix](#) [ReturnAsDenseMatrix](#) ([Real](#) west, [Real](#) east, int num\_cells\_x) const
- *Returns the operator as a dense matrix.*
- [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid) const
- *Returns the operator as a dense matrix.*
- [DenseMatrix](#) [ReturnAsDimensionlessDenseMatrix](#) (int num\_cells\_x) const
- *Returns the operator as a dimensionless dense matrix.*

## Private Member Functions

- bool [ComputeStencilInteriorGrid](#) (void)
- *Stage 1 of the CBS Algorithm.*
- bool [ComputeRationalBasisNullSpace](#) (void)
- *Stage 2.1 of the CBS Algorithm.*
- bool [ComputePreliminaryApproximations](#) (void)
- *Stage 2.2 of the CBS Algorithm.*
- bool [ComputeWeights](#) (void)
- *Stage 2.3 of the CBS Algorithm.*
- bool [ComputeStencilBoundaryGrid](#) (void)
- *Stage 2.4 of the CBS Algorithm.*
- bool [AssembleOperator](#) (void)
- *Stage 3 of the CBS Algorithm.*

## Private Attributes

- int [order\\_accuracy\\_](#)
- *Order of numerical accuracy of the operator.*
- int [dim\\_null\\_](#)
- *Dim. null-space for boundary approximations.*
- int [num\\_bndy\\_approxs\\_](#)

- Req. approximations at and near the boundary.*
- int [num\\_bndy\\_coeffs\\_](#)  
*Req. coeffs. per bndy pt. uni. order accuracy.*
- int [gradient\\_length\\_](#)  
*Length of the output array.*
- int [minrow\\_](#)  
*Row from the optimizer with the minimum rel. nor.*
- int [row\\_](#)  
*Row currently processed by the optimizer.*
- [DenseMatrix](#) [rat\\_basis\\_null\\_space\\_](#)  
*Rational b. null-space w. bndy.*
- [Real](#) \* [coeffs\\_interior\\_](#)  
*Interior stencil.*
- [Real](#) \* [prem\\_apps\\_](#)  
*2D array of boundary preliminary approximations.*
- [Real](#) \* [weights\\_crs\\_](#)  
*Array containing weights from CRSA.*
- [Real](#) \* [weights\\_cbs\\_](#)  
*Array containing weights from CBSA.*
- [Real](#) \* [mim\\_bndy\\_](#)  
*Array containing mimetic boundary approximations.*
- [Real](#) \* [gradient\\_](#)  
*Output array containing the operator and weights.*
- [Real](#) [mimetic\\_threshold\\_](#)  
*< Mimetic threshold.*

## Friends

- [std::ostream](#) & [operator<<](#) ([std::ostream](#) &stream, [Grad1D](#) &in)  
*Output stream operator for printing.*

## 17.8.1 Detailed Description

This class implements a 1D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Definition at line 81 of file [mtk\\_grad\\_1d.h](#).

## 17.8.2 Constructor & Destructor Documentation

### 17.8.2.1 mtk::Grad1D::Grad1D ( )

Definition at line 134 of file [mtk\\_grad\\_1d.cc](#).

### 17.8.2.2 mtk::Grad1D::Grad1D ( const [Grad1D](#) &grad )

## Parameters

<i>in</i>	<i>div</i>	Given divergence.
-----------	------------	-------------------

Definition at line 150 of file [mtk\\_grad\\_1d.cc](#).

### 17.8.2.3 mtk::Grad1D::~~Grad1D ( )

Definition at line 166 of file [mtk\\_grad\\_1d.cc](#).

## 17.8.3 Member Function Documentation

### 17.8.3.1 bool mtk::Grad1D::AssembleOperator ( void ) [private]

Construct the output array with the operator and its weights.

1. The first entry of the array will contain the order of accuracy.
2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.
3. The third entry will contain the collection of weights.
4. The next `dim_null + 1` entries will contain the collections of approximating coefficients for the west boundary of the grid.

Definition at line 1547 of file [mtk\\_grad\\_1d.cc](#).

### 17.8.3.2 mtk::Real \* mtk::Grad1D::coeffs\_interior ( ) const

## Returns

Coefficients for the interior of the grid.

Definition at line 331 of file [mtk\\_grad\\_1d.cc](#).

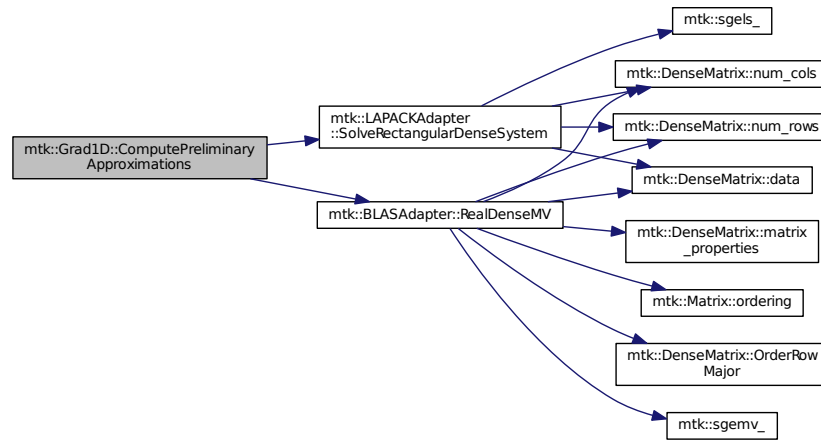
### 17.8.3.3 bool mtk::Grad1D::ComputePreliminaryApproximations ( void ) [private]

Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.
2. Compute the `dim_null` near-the-boundary columns of the pi matrix.
3. Create the Vandermonde matrix for this iteration.
4. New order-selector vector (gets re-written with LAPACK solutions).
5. Solving  $TT*rr = ob$  yields the columns `rr` of the `kk` matrix.
6. Scale the `kk` matrix to make it a rational basis for null-space.
7. Extract the last `dim_null` values of the pre-scaled `ob`.
8. Once we possess the bottom elements, we proceed with the scaling.

Definition at line 836 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



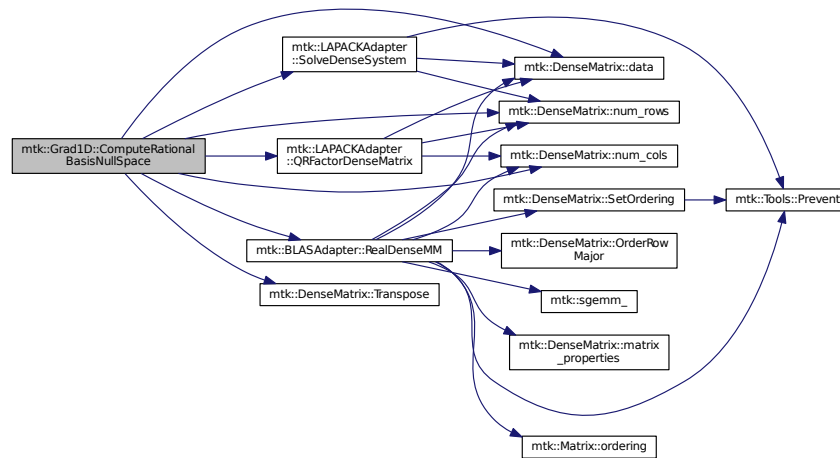
#### 17.8.3.4 `bool mtk::Grad1D::ComputeRationalBasisNullSpace ( void ) [private]`

Compute a rational basis for the null-space of the Vandermonde matrix approximating at the west boundary.

1. Create generator vector for the first approximation.
2. Create Vandermonde matrix.
3. QR-factorize the Vandermonde matrix.
4. Extract the basis for the null-space from Q matrix.
5. Scale null-space to make it rational.

Definition at line 653 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



#### 17.8.3.5 `bool mtk::Grad1D::ComputeStencilBoundaryGrid ( void ) [private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.
2. Compute alpha values.
3. Compute the mimetic boundary approximations.

Definition at line 1441 of file [mtk\\_grad\\_1d.cc](#).

#### 17.8.3.6 `bool mtk::Grad1D::ComputeStencilInteriorGrid ( void ) [private]`

Compute the stencil approximating the interior of the staggered grid.

1. Create vector for interior spatial coordinates.
2. Create Vandermonde matrix (using interior coordinates as generator).
3. Create order-selector vector.
4. Solve dense Vandermonde system to attain the interior coefficients.

Definition at line 556 of file [mtk\\_grad\\_1d.cc](#).



Here is the call graph for this function:



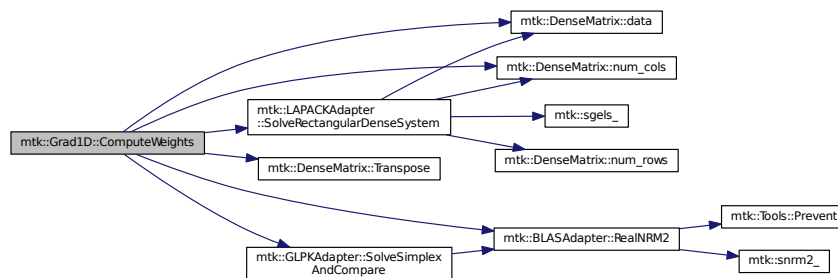
### 17.8.3.7 bool mtk::Grad1D::ComputeWeights ( void ) [private]

Compute the set of mimetic weights to impose the mimetic condition.

1. Construct the  $\mathbf{A}$  matrix.
2. Use interior stencil to build proper RHS vector  $\mathbf{h}$ .
3. Get weights (as **CRSA**):  $\mathbf{A}\mathbf{q} = \mathbf{h}$ .
4. If required order is greater than critical order, start the **CBSA**.
5. Create  $\mathbf{A}$  matrix from  $\mathbf{A}$ .
6. Prepare constraint vector as in the CBSA:  $\mathbf{c}$ .
7. Brute force search through all the rows of the  $\Phi$  matrix.
8. Apply solution found from brute force search.

Definition at line 1057 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



### 17.8.3.8 bool mtk::Grad1D::ConstructGrad1D ( int order\_accuracy = kDefaultOrderAccuracy, Real mimetic\_threshold = kDefaultMimeticThreshold )

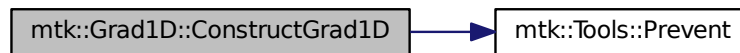
**Returns**

Success of the solution.

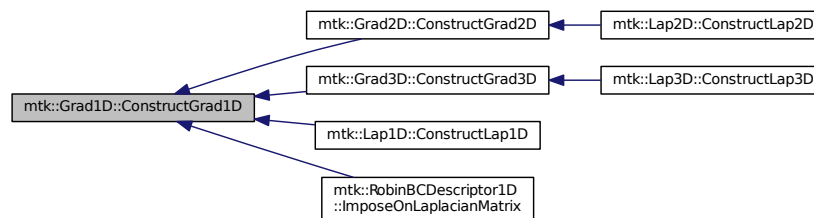
1. Compute stencil for the interior cells.
2. Compute a rational null-space from the first matrix transposed.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.
6. Assemble operator.

Definition at line 187 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.8.3.9 `mtk::DenseMatrix mtk::Grad1D::mim_bndy ( ) const`

**Returns**

Collection of mimetic approximations at the boundary.

Definition at line 346 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 17.8.3.10 `int mtk::Grad1D::num_bndy_coeffs ( ) const`

**Returns**

How many coefficients are approximating at the boundary.

Definition at line 326 of file [mtk\\_grad\\_1d.cc](#).

#### 17.8.3.11 `mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix ( mtk::Real west, mtk::Real east, int num_cells_x ) const`

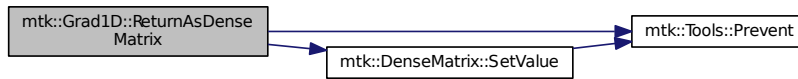
**Returns**

The operator as a dense matrix.

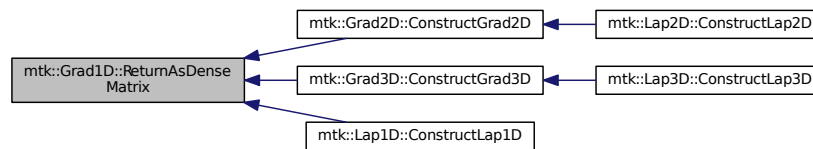
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 361 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 17.8.3.12 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix ( const UniStgGrid1D & grid ) const

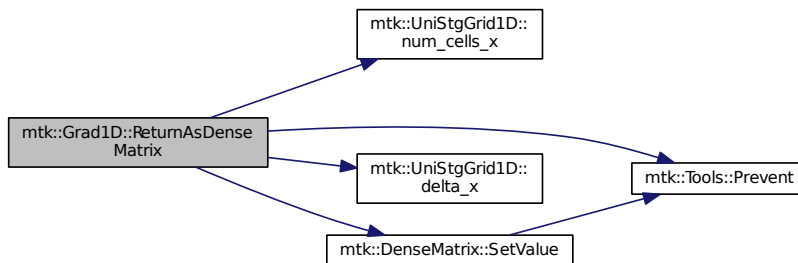
##### Returns

The operator as a dense matrix.

1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 430 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



### 17.8.3.13 mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix ( int *num\_cells\_x* ) const

#### Returns

The operator as a dimensionless dense matrix.

1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 494 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



### 17.8.3.14 mtk::Real \* mtk::Grad1D::weights\_cbs ( void ) const

#### Returns

Collection of weights as computed by the CBSA.

Definition at line 341 of file [mtk\\_grad\\_1d.cc](#).

### 17.8.3.15 mtk::Real \* mtk::Grad1D::weights\_crs ( void ) const

#### Returns

Success of the solution.

Definition at line 336 of file [mtk\\_grad\\_1d.cc](#).

## 17.8.4 Friends And Related Function Documentation

### 17.8.4.1 std::ostream& operator<< ( std::ostream & *stream*, mtk::Grad1D & *in* ) [friend]

1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

Definition at line 84 of file [mtk\\_grad\\_1d.cc](#).

### 17.8.5 Member Data Documentation

17.8.5.1 **Real\*** mtk::Grad1D::coeffs\_interior\_ [private]

Definition at line 217 of file [mtk\\_grad\\_1d.h](#).

17.8.5.2 **int** mtk::Grad1D::dim\_null\_ [private]

Definition at line 208 of file [mtk\\_grad\\_1d.h](#).

17.8.5.3 **Real\*** mtk::Grad1D::gradient\_ [private]

Definition at line 222 of file [mtk\\_grad\\_1d.h](#).

17.8.5.4 **int** mtk::Grad1D::gradient\_length\_ [private]

Definition at line 211 of file [mtk\\_grad\\_1d.h](#).

17.8.5.5 **Real\*** mtk::Grad1D::mim\_bndy\_ [private]

Definition at line 221 of file [mtk\\_grad\\_1d.h](#).

17.8.5.6 **Real** mtk::Grad1D::mimetic\_threshold\_ [private]

Definition at line 224 of file [mtk\\_grad\\_1d.h](#).

17.8.5.7 **int** mtk::Grad1D::minrow\_ [private]

Definition at line 212 of file [mtk\\_grad\\_1d.h](#).

17.8.5.8 **int** mtk::Grad1D::num\_bndy\_approxs\_ [private]

Definition at line 209 of file [mtk\\_grad\\_1d.h](#).

17.8.5.9 **int** mtk::Grad1D::num\_bndy\_coeffs\_ [private]

Definition at line 210 of file [mtk\\_grad\\_1d.h](#).

17.8.5.10 **int** mtk::Grad1D::order\_accuracy\_ [private]

Definition at line 207 of file [mtk\\_grad\\_1d.h](#).

17.8.5.11 **Real\*** mtk::Grad1D::prem\_apps\_ [private]

Definition at line 218 of file [mtk\\_grad\\_1d.h](#).

17.8.5.12 **DenseMatrix** mtk::Grad1D::rat\_basis\_null\_space\_ [private]

Definition at line 215 of file [mtk\\_grad\\_1d.h](#).

17.8.5.13 **int** mtk::Grad1D::row\_ [private]

Definition at line 213 of file [mtk\\_grad\\_1d.h](#).

17.8.5.14 **Real\*** mtk::Grad1D::weights\_cbs\_ [private]

Definition at line 220 of file [mtk\\_grad\\_1d.h](#).

17.8.5.15 **Real\*** mtk::Grad1D::weights\_crs\_ [private]

Definition at line 219 of file [mtk\\_grad\\_1d.h](#).

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

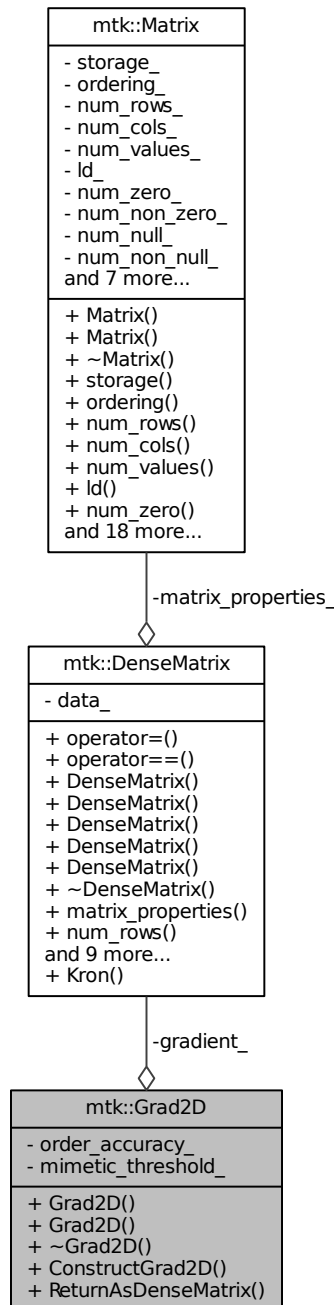
- [include/mtk\\_grad\\_1d.h](#)
- [src/mtk\\_grad\\_1d.cc](#)

## 17.9 mtk::Grad2D Class Reference

Implements a 2D mimetic gradient operator.

```
#include <mtk_grad_2d.h>
```

Collaboration diagram for mtk::Grad2D:



## Public Member Functions

- [Grad2D \(\)](#)



*Default constructor.*

- [Grad2D](#) (const [Grad2D](#) &grad)

*Copy constructor.*

- [~Grad2D](#) ()

*Destructor.*

- bool [ConstructGrad2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_↔ threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix](#) gradient\_

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real](#) [mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.9.1 Detailed Description

This class implements a 2D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Definition at line 76 of file [mtk\\_grad\\_2d.h](#).

## 17.9.2 Constructor & Destructor Documentation

### 17.9.2.1 mtk::Grad2D::Grad2D ( )

Definition at line 67 of file [mtk\\_grad\\_2d.cc](#).

### 17.9.2.2 mtk::Grad2D::Grad2D ( const Grad2D & grad )

#### Parameters

<a href="#">in</a>	<a href="#">div</a>	Given divergence.
--------------------	---------------------	-------------------

Definition at line 71 of file [mtk\\_grad\\_2d.cc](#).

### 17.9.2.3 mtk::Grad2D::~~Grad2D ( )

Definition at line 75 of file [mtk\\_grad\\_2d.cc](#).

### 17.9.3 Member Function Documentation

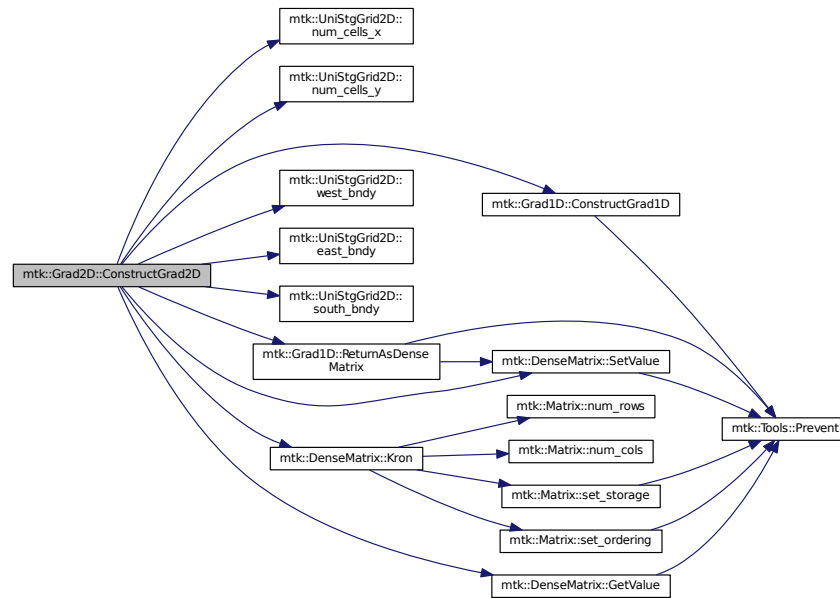
17.9.3.1 `bool mtk::Grad2D::ConstructGrad2D ( const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

Definition at line 77 of file [mtk\\_grad\\_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.9.3.2 `mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix ( ) const`

**Returns**

The operator as a dense matrix.

Definition at line 145 of file [mtk\\_grad\\_2d.cc](#).

Here is the caller graph for this function:

**17.9.4 Member Data Documentation****17.9.4.1 DenseMatrix mtk::Grad2D::gradient\_ [private]**

Definition at line 108 of file [mtk\\_grad\\_2d.h](#).

**17.9.4.2 Real mtk::Grad2D::mimetic\_threshold\_ [private]**

Definition at line 112 of file [mtk\\_grad\\_2d.h](#).

**17.9.4.3 int mtk::Grad2D::order\_accuracy\_ [private]**

Definition at line 110 of file [mtk\\_grad\\_2d.h](#).

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

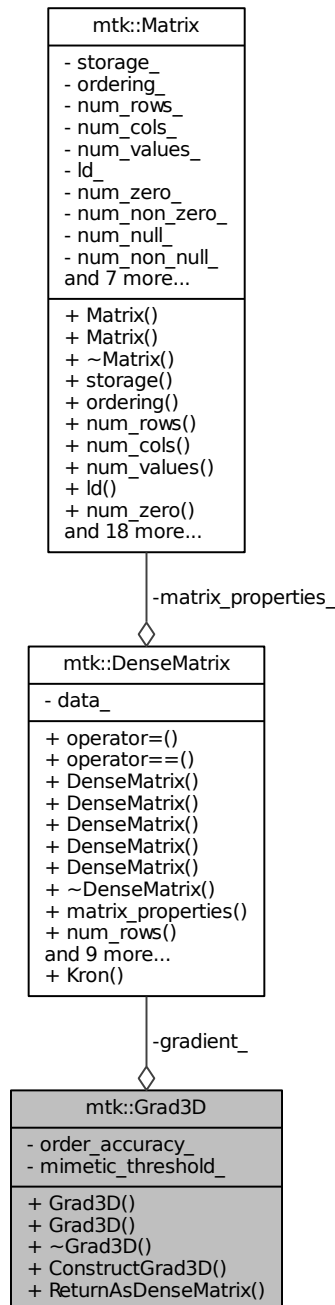
- [include/mtk\\_grad\\_2d.h](#)
- [src/mtk\\_grad\\_2d.cc](#)

**17.10 mtk::Grad3D Class Reference**

Implements a 3D mimetic gradient operator.

```
#include <mtk_grad_3d.h>
```

Collaboration diagram for mtk::Grad3D:



## Public Member Functions

- [Grad3D \(\)](#)

*Default constructor.*

- [Grad3D](#) (const [Grad3D](#) &grad)

*Copy constructor.*

- [~Grad3D](#) ()

*Destructor.*

- bool [ConstructGrad3D](#) (const [UniStgGrid3D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_↔ threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix](#) gradient\_

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real](#) [mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.10.1 Detailed Description

This class implements a 3D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Definition at line 76 of file [mtk\\_grad\\_3d.h](#).

## 17.10.2 Constructor & Destructor Documentation

### 17.10.2.1 mtk::Grad3D::Grad3D ( )

Definition at line 67 of file [mtk\\_grad\\_3d.cc](#).

### 17.10.2.2 mtk::Grad3D::Grad3D ( const Grad3D & grad )

#### Parameters

<a href="#">in</a>	<a href="#">div</a>	Given divergence.
--------------------	---------------------	-------------------

Definition at line 71 of file [mtk\\_grad\\_3d.cc](#).

### 17.10.2.3 mtk::Grad3D::~~Grad3D ( )

Definition at line 75 of file [mtk\\_grad\\_3d.cc](#).

### 17.10.3 Member Function Documentation

17.10.3.1 `bool mtk::Grad3D::ConstructGrad3D ( const UniStgGrid3D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

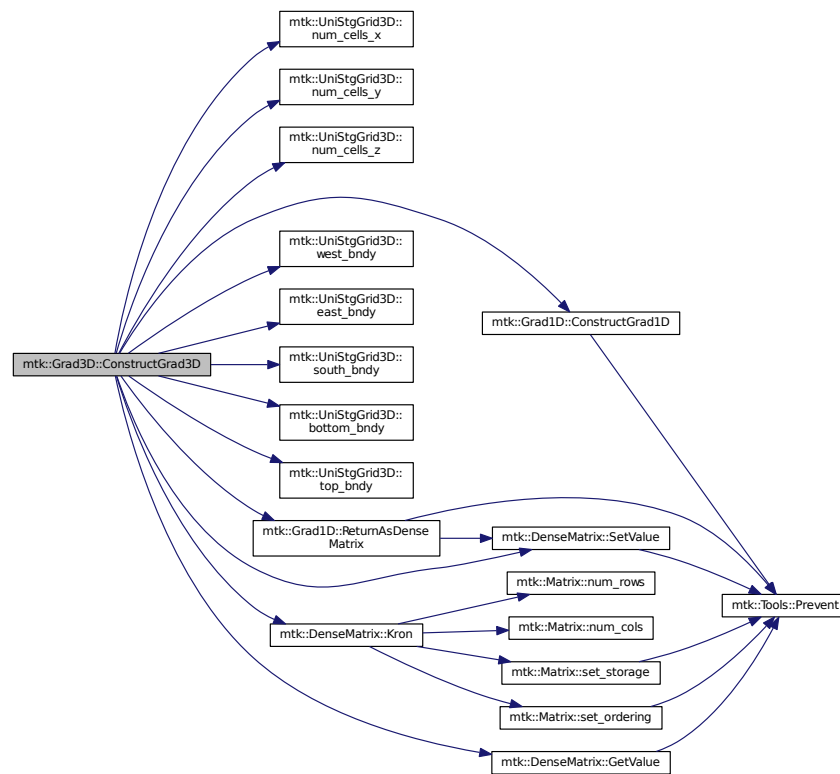
#### Returns

Success of the construction.

1. Build preliminary staggering through the x direction.
2. Build preliminary staggering through the y direction.
3. Build preliminary staggering through the z direction.
4. Actual operator:  $GG_{xyz} = [gx; gy; gz]$ .

Definition at line 77 of file [mtk\\_grad\\_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 17.10.3.2 `mtk::DenseMatrix mtk::Grad3D::ReturnAsDenseMatrix ( ) const`

##### Returns

The operator as a dense matrix.

Definition at line 185 of file [mtk\\_grad\\_3d.cc](#).

Here is the caller graph for this function:



### 17.10.4 Member Data Documentation

#### 17.10.4.1 `DenseMatrix mtk::Grad3D::gradient_ [private]`

Definition at line 108 of file [mtk\\_grad\\_3d.h](#).

#### 17.10.4.2 `Real mtk::Grad3D::mimetic_threshold_ [private]`

Definition at line 112 of file [mtk\\_grad\\_3d.h](#).

#### 17.10.4.3 `int mtk::Grad3D::order_accuracy_ [private]`

Definition at line 110 of file [mtk\\_grad\\_3d.h](#).

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

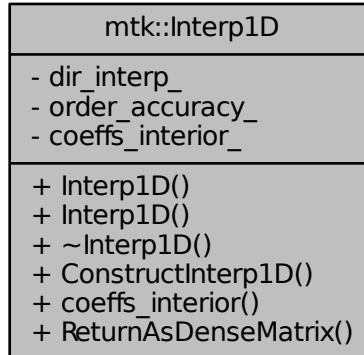
- [include/mtk\\_grad\\_3d.h](#)
- [src/mtk\\_grad\\_3d.cc](#)

## 17.11 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

```
#include <mtk_interp_1d.h>
```

Collaboration diagram for mtk::Interp1D:



### Public Member Functions

- [Interp1D \(\)](#)  
*Default constructor.*
- [Interp1D \(const \[Interp1D\]\(#\) &interp\)](#)  
*Copy constructor.*
- [~Interp1D \(\)](#)  
*Destructor.*
- [bool ConstructInterp1D \(int order\\_accuracy=kDefaultOrderAccuracy, mtk::DirInterp dir=SCALAR\\_TO\\_VECTOR\)](#)  
*Factory method to build operator.*
- [Real \\* coeffs\\_interior \(\) const](#)  
*Returns coefficients for the interior of the grid.*
- [DenseMatrix ReturnAsDenseMatrix \(const \[UniStgGrid1D\]\(#\) &grid\) const](#)  
*Returns the operator as a dense matrix.*

### Private Attributes

- [DirInterp dir\\_interp\\_](#)  
*Direction of interpolation.*
- [int order\\_accuracy\\_](#)  
*Order of numerical accuracy of the operator.*
- [Real \\* coeffs\\_interior\\_](#)  
*Interior stencil.*



## Friends

- `std::ostream & operator<< (std::ostream &stream, Interp1D &in)`  
Output stream operator for printing.

### 17.11.1 Detailed Description

This class implements a 1D interpolation operator.

Definition at line 82 of file [mtk\\_interp\\_1d.h](#).

### 17.11.2 Constructor & Destructor Documentation

#### 17.11.2.1 `mtk::Interp1D::Interp1D ( )`

Definition at line 80 of file [mtk\\_interp\\_1d.cc](#).

#### 17.11.2.2 `mtk::Interp1D::Interp1D ( const Interp1D &interp )`

##### Parameters

<code>in</code>	<code>interp</code>	Given interpolation operator.
-----------------	---------------------	-------------------------------

Definition at line 85 of file [mtk\\_interp\\_1d.cc](#).

#### 17.11.2.3 `mtk::Interp1D::~~Interp1D ( )`

Definition at line 90 of file [mtk\\_interp\\_1d.cc](#).

### 17.11.3 Member Function Documentation

#### 17.11.3.1 `mtk::Real * mtk::Interp1D::coeffs_interior ( ) const`

##### Returns

Coefficients for the interior of the grid.

Definition at line 132 of file [mtk\\_interp\\_1d.cc](#).

#### 17.11.3.2 `bool mtk::Interp1D::ConstructInterp1D ( int order_accuracy = kDefaultOrderAccuracy, mtk::DirInterp dir = SCALAR_TO_VECTOR )`

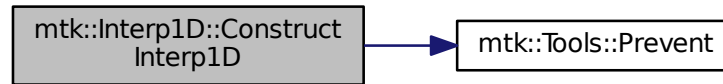
##### Returns

Success of the solution.

1. Compute stencil for the interior cells.

Definition at line 96 of file [mtk\\_interp\\_1d.cc](#).

Here is the call graph for this function:



### 17.11.3.3 `mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix ( const UniStgGrid1D & grid ) const`

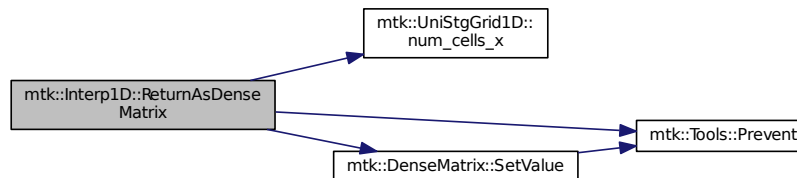
Returns

The operator as a dense matrix.

1. Preserve values at the boundary.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the boundaries.

Definition at line 137 of file [mtk\\_interp\\_1d.cc](#).

Here is the call graph for this function:



## 17.11.4 Friends And Related Function Documentation

### 17.11.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::Interp1D & in ) [friend]`

1. Print approximating coefficients for the interior.

Definition at line 66 of file [mtk\\_interp\\_1d.cc](#).

## 17.11.5 Member Data Documentation

### 17.11.5.1 `Real* mtk::Interp1D::coeffs_interior_ [private]`

Definition at line 127 of file [mtk\\_interp\\_1d.h](#).

#### 17.11.5.2 DirInterp mtk::Interp1D::dir\_interp\_ [private]

Definition at line 123 of file [mtk\\_interp\\_1d.h](#).

#### 17.11.5.3 int mtk::Interp1D::order\_accuracy\_ [private]

Definition at line 125 of file [mtk\\_interp\\_1d.h](#).

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

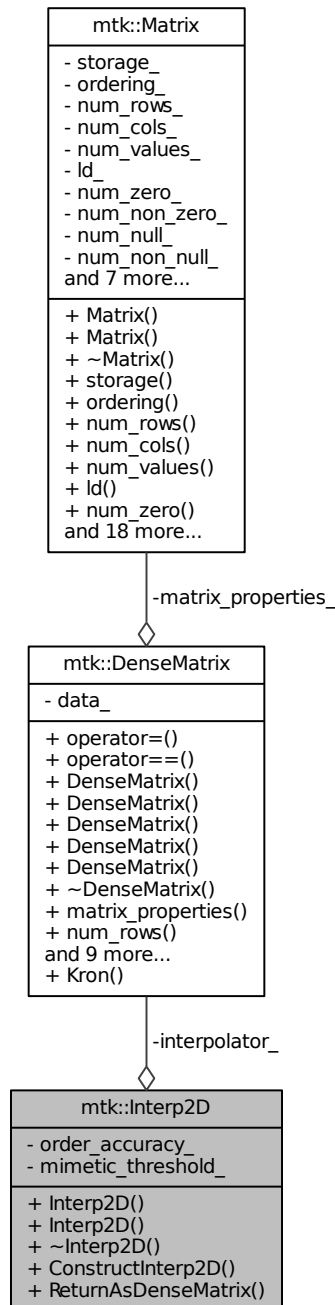
- [include/mtk\\_interp\\_1d.h](#)
- [src/mtk\\_interp\\_1d.cc](#)

## 17.12 mtk::Interp2D Class Reference

Implements a 2D interpolation operator.

```
#include <mtk_interp_2d.h>
```

Collaboration diagram for mtk::Interp2D:



## Public Member Functions

- [Interp2D \(\)](#)

*Default constructor.*

- [Interp2D](#) (const [Interp2D](#) &interp)

*Copy constructor.*

- [~Interp2D](#) ()

*Destructor.*

- [DenseMatrix ConstructInterp2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) ()

*Return the operator as a dense matrix.*

## Private Attributes

- [DenseMatrix interpolator\\_](#)

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

## 17.12.1 Detailed Description

This class implements a 2D interpolation operator.

Definition at line 76 of file [mtk\\_interp\\_2d.h](#).

## 17.12.2 Constructor & Destructor Documentation

17.12.2.1 [mtk::Interp2D::Interp2D](#) ( )

17.12.2.2 [mtk::Interp2D::Interp2D](#) ( const [Interp2D](#) & *interp* )

Parameters

<a href="#">in</a>	<a href="#">lap</a>	Given Laplacian.
--------------------	---------------------	------------------

17.12.2.3 [mtk::Interp2D::~~Interp2D](#) ( )

## 17.12.3 Member Function Documentation

17.12.3.1 [DenseMatrix mtk::Interp2D::ConstructInterp2D](#) ( const [UniStgGrid2D](#) & *grid*, int *order\_accuracy* = [kDefaultOrderAccuracy](#), [Real](#) *mimetic\_threshold* = [kDefaultMimeticThreshold](#) )

Returns

Success of the construction.

#### 17.12.3.2 DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix ( )

##### Returns

The operator as a dense matrix.

### 17.12.4 Member Data Documentation

#### 17.12.4.1 DenseMatrix mtk::Interp2D::interpolator\_ [private]

Definition at line 108 of file [mtk\\_interp\\_2d.h](#).

#### 17.12.4.2 Real mtk::Interp2D::mimetic\_threshold\_ [private]

Definition at line 112 of file [mtk\\_interp\\_2d.h](#).

#### 17.12.4.3 int mtk::Interp2D::order\_accuracy\_ [private]

Definition at line 110 of file [mtk\\_interp\\_2d.h](#).

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

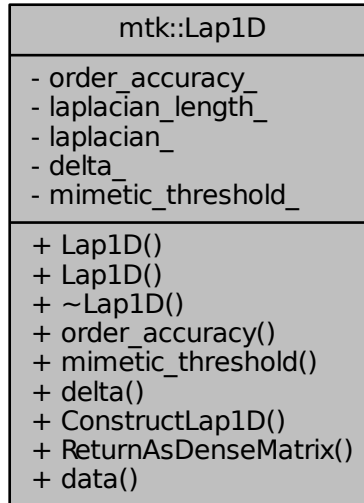
- [include/mtk\\_interp\\_2d.h](#)

## 17.13 mtk::Lap1D Class Reference

Implements a 1D mimetic Laplacian operator.

```
#include <mtk_lap_1d.h>
```

Collaboration diagram for mtk::Lap1D:



## Public Member Functions

- [Lap1D](#) ()  
*Default constructor.*
- [Lap1D](#) (const [Lap1D](#) &lap)  
*Copy constructor.*
- [~Lap1D](#) ()  
*Destructor.*
- int [order\\_accuracy](#) () const  
*Order of accuracy of the operator.*
- [Real](#) [mimetic\\_threshold](#) () const  
*Mimetic threshold used in the CBS algorithm to construct this operator.*
- [Real](#) [delta](#) () const  
*Value of  $\Delta x$  used be scaled. If 0, then dimensionless.*
- bool [ConstructLap1D](#) (int [order\\_accuracy](#)=kDefaultOrderAccuracy, [Real](#) [mimetic\\_threshold](#)=kDefaultMimeticThreshold)  
*Factory method implementing the CBS Algorithm to build operator.*
- [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid) const  
*Return the operator as a dense matrix.*
- const [mtk::Real](#) \* [data](#) (const [UniStgGrid1D](#) &grid) const  
*Return the operator as a dense array.*

## Private Attributes

- int [order\\_accuracy\\_](#)  
*Order of numerical accuracy of the operator.*
- int [laplacian\\_length\\_](#)  
*Length of the output array.*
- [Real](#) \* [laplacian\\_](#)  
*Output array containing the operator and weights.*
- [Real](#) [delta\\_](#)  
*< If 0.0, then this Laplacian is dimensionless.*
- [Real](#) [mimetic\\_threshold\\_](#)  
*< Mimetic threshold.*

## Friends

- [std::ostream](#) & [operator<<](#) ([std::ostream](#) &stream, [Lap1D](#) &in)  
*Output stream operator for printing.*

### 17.13.1 Detailed Description

This class implements a 1D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 76 of file [mtk\\_lap\\_1d.h](#).

### 17.13.2 Constructor & Destructor Documentation

#### 17.13.2.1 [mtk::Lap1D::Lap1D \( \)](#)

Definition at line 108 of file [mtk\\_lap\\_1d.cc](#).

#### 17.13.2.2 [mtk::Lap1D::Lap1D \( const Lap1D & lap \)](#)

##### Parameters

<a href="#">in</a>	<a href="#">lap</a>	Given Laplacian.
--------------------	---------------------	------------------

#### 17.13.2.3 [mtk::Lap1D::~~Lap1D \( \)](#)

Definition at line 114 of file [mtk\\_lap\\_1d.cc](#).

### 17.13.3 Member Function Documentation

#### 17.13.3.1 [bool mtk::Lap1D::ConstructLap1D \( int order\\_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic\\_threshold = kDefaultMimeticThreshold \)](#)



**Returns**

Success of the solution.

1. Create gradient operator using specific values for the Laplacian.
2. Create gradient operator using specific values for the Laplacian.
3. Create both operators as matrices.
4. Multiply both operators:  $\check{\mathbf{L}}_x^k = \check{\mathbf{D}}_x^k \check{\mathbf{G}}_x^k$
5. Extract the coefficients from the matrix and store them in the array.

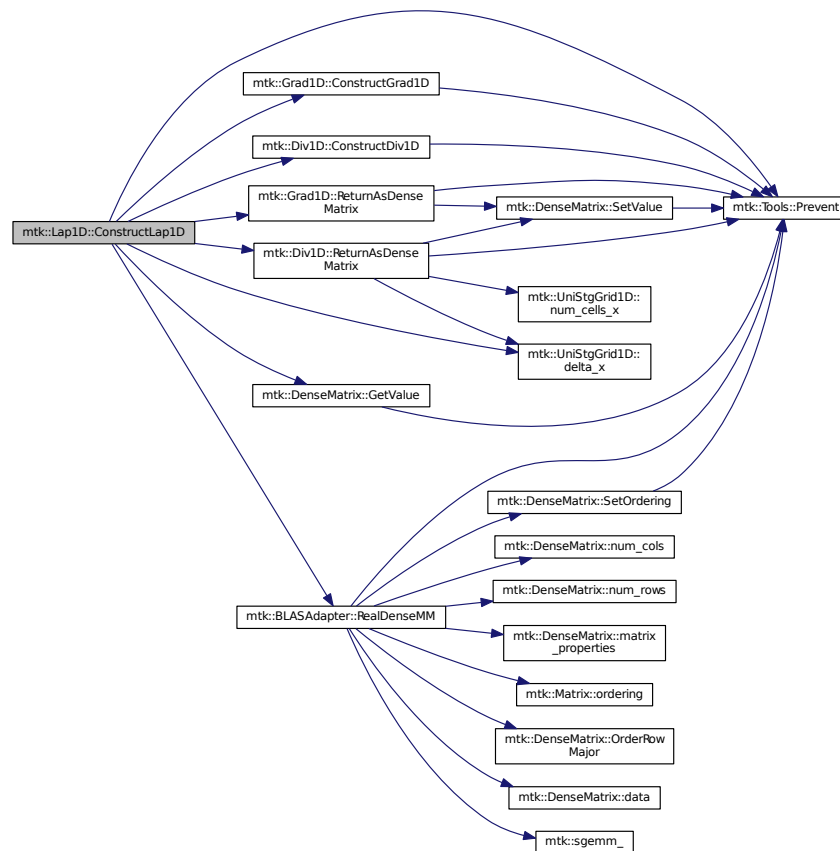
**Warning**

We do not compute weights for this operator... no need to!

1. The first entry of the array will contain the order of accuracy.
2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.
3. We DO NOT have weights in this operator. Copy mimetic bndy coeffs.

Definition at line 135 of file [mtk\\_lap\\_1d.cc](#).

Here is the call graph for this function:



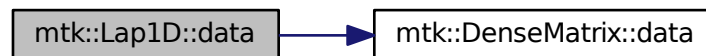
17.13.3.2 `const mtk::Real * mtk::Lap1D::data ( const UniStgGrid1D & grid ) const`

#### Returns

The operator as a dense array.

Definition at line 356 of file [mtk\\_lap\\_1d.cc](#).

Here is the call graph for this function:



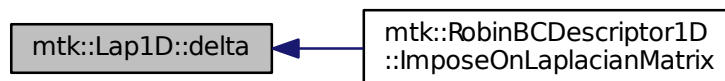
17.13.3.3 `mtk::Real mtk::Lap1D::delta ( ) const`

#### Returns

Value of  $\Delta x$  used be scaled. If 0, then dimensionless.

Definition at line 130 of file [mtk\\_lap\\_1d.cc](#).

Here is the caller graph for this function:



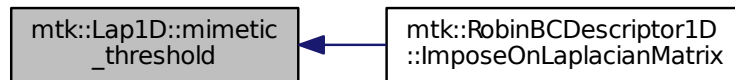
17.13.3.4 `mtk::Real mtk::Lap1D::mimetic_threshold ( ) const`

**Returns**

Mimetic threshold used in the CBS algorithm to construct operator.

Definition at line 125 of file [mtk\\_lap\\_1d.cc](#).

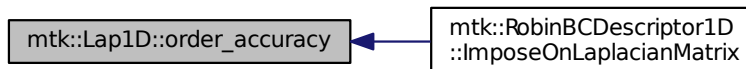
Here is the caller graph for this function:

**17.13.3.5 int mtk::Lap1D::order\_accuracy ( ) const****Returns**

Order of accuracy of the operator.

Definition at line 120 of file [mtk\\_lap\\_1d.cc](#).

Here is the caller graph for this function:

**17.13.3.6 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix ( const UniStgGrid1D & grid ) const****Returns**

The operator as a dense matrix.

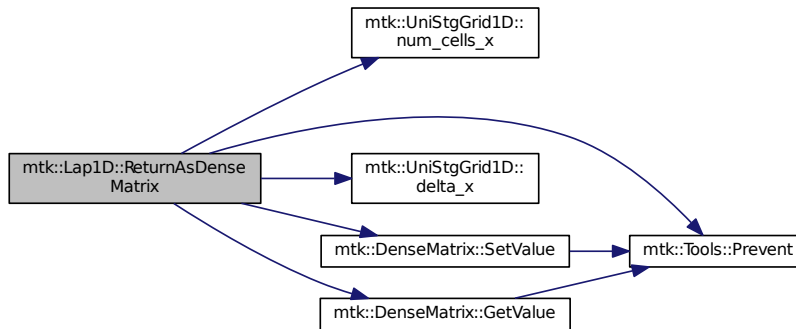
1. Extract mimetic coefficients from the west boundary.
2. Extract interior coefficients.
3. Extract mimetic coefficients from the west boundary to go east.

## Note

We could create two matrices of the requested size and multiply them, but that would be inefficient, since we already have the computed coefficients stored. We just have to set them in place, in a matrix of an adequate size, and multiply them times the inverse of the square of the step size, in order for the matrix to actually represent a differential operator.

Definition at line 286 of file [mtk\\_lap\\_1d.cc](#).

Here is the call graph for this function:



### 17.13.4 Friends And Related Function Documentation

17.13.4.1 `std::ostream& operator<<( std::ostream & stream, mtk::Lap1D & in )` [*friend*]

1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

Definition at line 73 of file [mtk\\_lap\\_1d.cc](#).

### 17.13.5 Member Data Documentation

17.13.5.1 `Real mtk::Lap1D::delta_` [*mutable*], [*private*]

Definition at line 143 of file [mtk\\_lap\\_1d.h](#).

17.13.5.2 `Real* mtk::Lap1D::laplacian_` [*private*]

Definition at line 141 of file [mtk\\_lap\\_1d.h](#).

17.13.5.3 `int mtk::Lap1D::laplacian_length_` [*private*]

Definition at line 139 of file [mtk\\_lap\\_1d.h](#).

#### 17.13.5.4 Real mtk::Lap1D::mimetic\_threshold\_ [private]

Definition at line 145 of file [mtk\\_lap\\_1d.h](#).

#### 17.13.5.5 int mtk::Lap1D::order\_accuracy\_ [private]

Definition at line 138 of file [mtk\\_lap\\_1d.h](#).

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

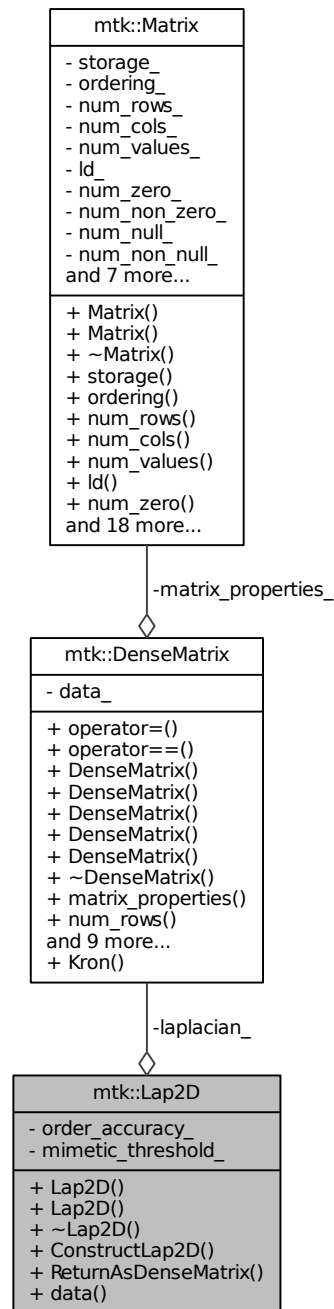
- [include/mtk\\_lap\\_1d.h](#)
- [src/mtk\\_lap\\_1d.cc](#)

## 17.14 mtk::Lap2D Class Reference

Implements a 2D mimetic Laplacian operator.

```
#include <mtk_lap_2d.h>
```

Collaboration diagram for mtk::Lap2D:



## Public Member Functions

- [Lap2D \(\)](#)

*Default constructor.*

- [Lap2D](#) (const [Lap2D](#) &lap)

*Copy constructor.*

- [~Lap2D](#) ()

*Destructor.*

- bool [ConstructLap2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_↔ threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix](#) [ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

- [Real](#) \* [data](#) () const

*Return the operator as a dense array.*

## Private Attributes

- [DenseMatrix](#) [laplacian\\_](#)

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real](#) [mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

### 17.14.1 Detailed Description

This class implements a 2D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 76 of file [mtk\\_lap\\_2d.h](#).

### 17.14.2 Constructor & Destructor Documentation

#### 17.14.2.1 mtk::Lap2D::Lap2D ( )

Definition at line 69 of file [mtk\\_lap\\_2d.cc](#).

#### 17.14.2.2 mtk::Lap2D::Lap2D ( const Lap2D & lap )

##### Parameters

<a href="#">in</a>	<a href="#">lap</a>	Given Laplacian.
--------------------	---------------------	------------------

Definition at line 71 of file [mtk\\_lap\\_2d.cc](#).

#### 17.14.2.3 mtk::Lap2D::~~Lap2D ( )

Definition at line 75 of file [mtk\\_lap\\_2d.cc](#).

### 17.14.3 Member Function Documentation

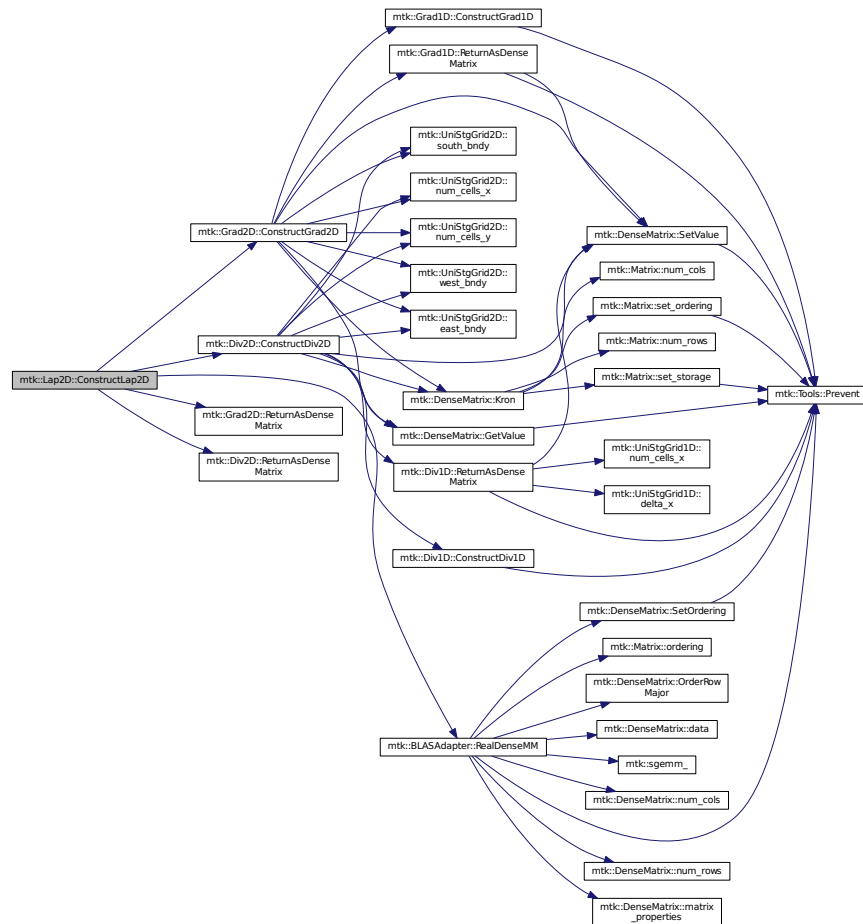
17.14.3.1 `bool mtk::Lap2D::ConstructLap2D ( const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

Definition at line 77 of file [mtk\\_lap\\_2d.cc](#).

Here is the call graph for this function:



17.14.3.2 `mtk::Real * mtk::Lap2D::data ( ) const`

#### Returns

The operator as a dense array.

Definition at line 115 of file [mtk\\_lap\\_2d.cc](#).



17.14.3.3 **mtk::DenseMatrix** mtk::Lap2D::ReturnAsDenseMatrix ( ) const

#### Returns

The operator as a dense matrix.

Definition at line 110 of file [mtk\\_lap\\_2d.cc](#).

### 17.14.4 Member Data Documentation

17.14.4.1 **DenseMatrix** mtk::Lap2D::laplacian\_ [private]

Definition at line 115 of file [mtk\\_lap\\_2d.h](#).

17.14.4.2 **Real** mtk::Lap2D::mimetic\_threshold\_ [private]

Definition at line 119 of file [mtk\\_lap\\_2d.h](#).

17.14.4.3 **int** mtk::Lap2D::order\_accuracy\_ [private]

Definition at line 117 of file [mtk\\_lap\\_2d.h](#).

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

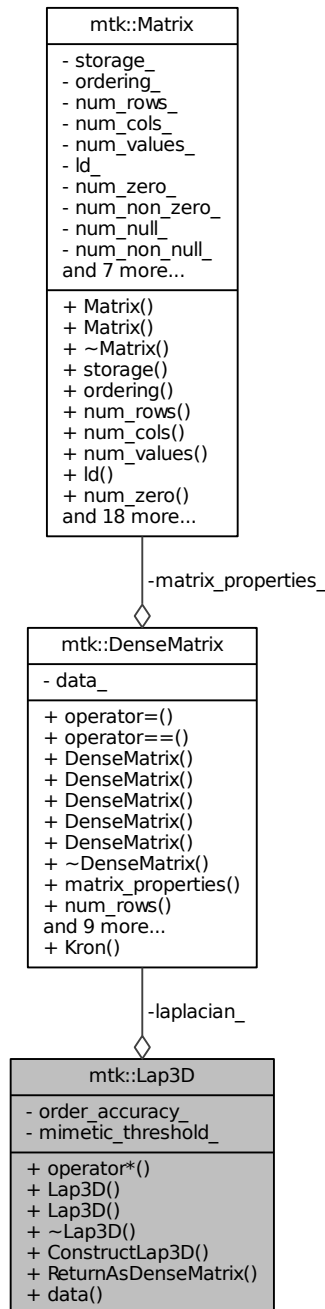
- [include/mtk\\_lap\\_2d.h](#)
- [src/mtk\\_lap\\_2d.cc](#)

## 17.15 mtk::Lap3D Class Reference

Implements a 3D mimetic Laplacian operator.

```
#include <mtk_lap_3d.h>
```

Collaboration diagram for mtk::Lap3D:



## Public Member Functions

- [UniStgGrid3D operator\\*](#) (const [UniStgGrid3D](#) &grid) const

*Operator application operator on a grid.*

- [Lap3D](#) ()

*Default constructor.*

- [Lap3D](#) (const [Lap3D](#) &lap)

*Copy constructor.*

- [~Lap3D](#) ()

*Destructor.*

- bool [ConstructLap3D](#) (const [UniStgGrid3D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_↵ threshold=[kDefaultMimeticThreshold](#))

*Factory method implementing the CBS Algorithm to build operator.*

- [DenseMatrix ReturnAsDenseMatrix](#) () const

*Return the operator as a dense matrix.*

- [Real \\* data](#) () const

*Return the operator as a dense array.*

## Private Attributes

- [DenseMatrix laplacian\\_](#)

*Actual operator.*

- int [order\\_accuracy\\_](#)

*Order of accuracy.*

- [Real mimetic\\_threshold\\_](#)

*Mimetic Threshold.*

### 17.15.1 Detailed Description

This class implements a 3D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Definition at line 76 of file [mtk\\_lap\\_3d.h](#).

### 17.15.2 Constructor & Destructor Documentation

#### 17.15.2.1 mtk::Lap3D::Lap3D ( )

Definition at line 76 of file [mtk\\_lap\\_3d.cc](#).

#### 17.15.2.2 mtk::Lap3D::Lap3D ( const Lap3D & lap )

##### Parameters

<a href="#">in</a>	<a href="#">lap</a>	Given Laplacian.
--------------------	---------------------	------------------

Definition at line 78 of file [mtk\\_lap\\_3d.cc](#).

#### 17.15.2.3 mtk::Lap3D::~~Lap3D ( )

Definition at line 82 of file [mtk\\_lap\\_3d.cc](#).

### 17.15.3 Member Function Documentation

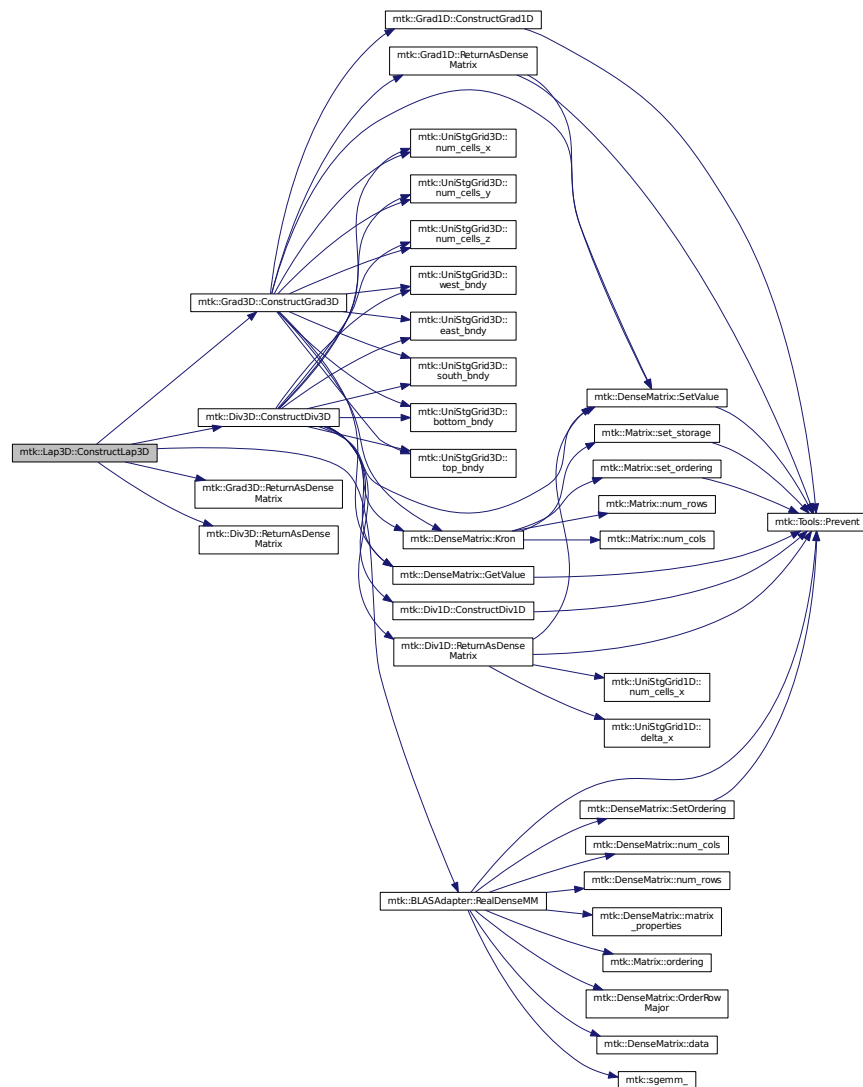
17.15.3.1 `bool mtk::Lap3D::ConstructLap3D ( const UniStgGrid3D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold )`

#### Returns

Success of the construction.

Definition at line 84 of file `mtk_lap_3d.cc`.

Here is the call graph for this function:



17.15.3.2 `mtk::Real * mtk::Lap3D::data ( ) const`

**Returns**

The operator as a dense array.

Definition at line 122 of file [mtk\\_lap\\_3d.cc](#).

**17.15.3.3** `mtk::UniStgGrid3D mtk::Lap3D::operator* ( const UniStgGrid3D & grid ) const`

Definition at line 69 of file [mtk\\_lap\\_3d.cc](#).

**17.15.3.4** `mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix ( ) const`

**Returns**

The operator as a dense matrix.

Definition at line 117 of file [mtk\\_lap\\_3d.cc](#).

**17.15.4 Member Data Documentation**

**17.15.4.1** `DenseMatrix mtk::Lap3D::laplacian_ [private]`

Definition at line 118 of file [mtk\\_lap\\_3d.h](#).

**17.15.4.2** `Real mtk::Lap3D::mimetic_threshold_ [private]`

Definition at line 122 of file [mtk\\_lap\\_3d.h](#).

**17.15.4.3** `int mtk::Lap3D::order_accuracy_ [private]`

Definition at line 120 of file [mtk\\_lap\\_3d.h](#).

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

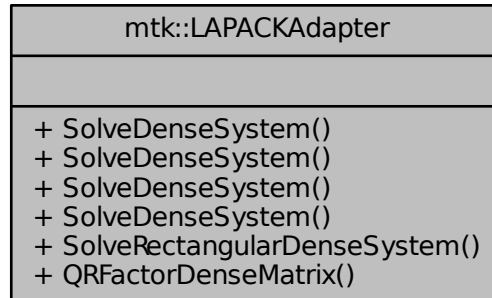
- [include/mtk\\_lap\\_3d.h](#)
- [src/mtk\\_lap\\_3d.cc](#)

**17.16 mtk::LAPACKAdapter Class Reference**

Adapter class for the LAPACK API.

```
#include <mtk_lapack_adapter.h>
```

Collaboration diagram for mtk::LAPACKAdapter:



### Static Public Member Functions

- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::Real](#) \*rhs)  
*Solves a dense system of linear equations.*
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::DenseMatrix](#) &rr)  
*Solves a dense system of linear equations.*
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid1D](#) &rhs)  
*Solves a dense system of linear equations.*
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid2D](#) &rhs)  
*Solves a dense system of linear equations.*
- static int [SolveRectangularDenseSystem](#) (const [mtk::DenseMatrix](#) &aa, [mtk::Real](#) \*ob\_, int ob\_Id\_)  
*Solves overdetermined or underdetermined real linear systems.*
- static [mtk::DenseMatrix](#) [QRFactorDenseMatrix](#) ([DenseMatrix](#) &matrix)  
*Performs a QR factorization on a dense matrix.*

### 17.16.1 Detailed Description

This class contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit the numerical methods implemented in the LAPACK.

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Definition at line 94 of file [mtk\\_lapack\\_adapter.h](#).

## 17.16.2 Member Function Documentation

17.16.2.1 `mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix ( mtk::DenseMatrix & aa ) [static]`

Adapts the MTK to LAPACK's routine.

## Parameters

<code>in, out</code>	<code>matrix</code>	Input matrix.
----------------------	---------------------	---------------

## Returns

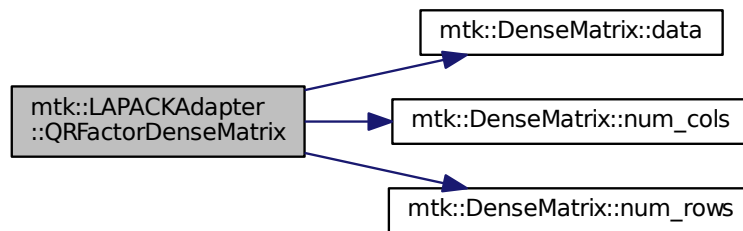
Matrix **Q**.

## Exceptions

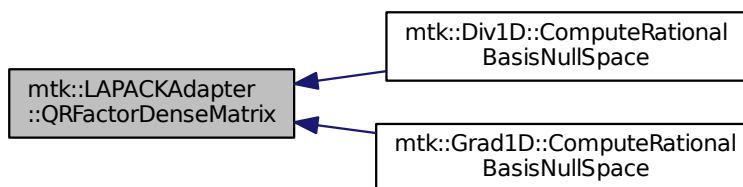
<code>std::bad_alloc</code>
-----------------------------

Definition at line 594 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



17.16.2.2 `int mtk::LAPACKAdapter::SolveDenseSystem ( mtk::DenseMatrix & mm, mtk::Real * rhs ) [static]`

Adapts the MTK to LAPACK's `dgesv_` routine.



## Parameters

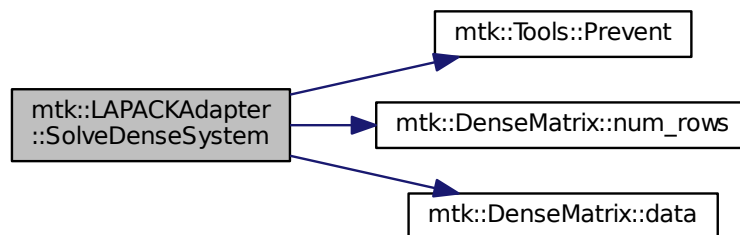
in	<i>matrix</i>	Input matrix.
in	<i>rhs</i>	Input right-hand sides vector.

## Exceptions

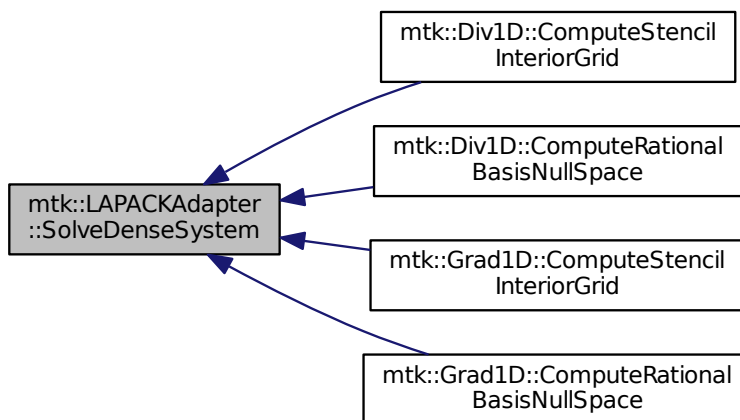
<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 431 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.16.2.3 `int mtk::LAPACKAdapter::SolveDenseSystem ( mtk::DenseMatrix & mm, mtk::DenseMatrix & rr ) [static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

## Parameters

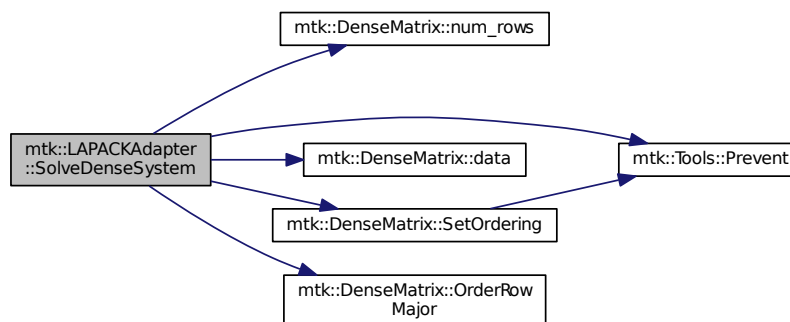
in	<i>matrix</i>	Input matrix.
in	<i>rr</i>	Input right-hand sides matrix.

## Exceptions

<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 466 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



17.16.2.4 `int mtk::LAPACKAdapter::SolveDenseSystem ( mtk::DenseMatrix & mm, mtk::UniStgGrid1D & rhs )`  
`[static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

## Parameters

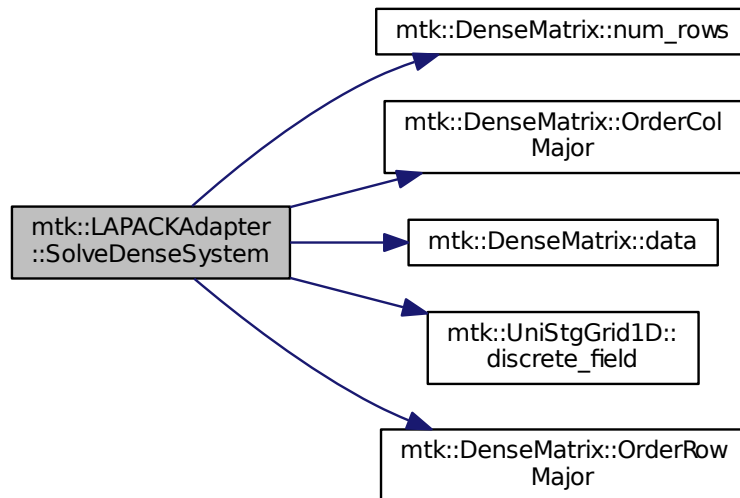
in	<i>matrix</i>	Input matrix.
in	<i>rhs</i>	Input right-hand side from info on a grid.

## Exceptions

<i>std::bad_alloc</i>	
-----------------------	--

Definition at line 518 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



17.16.2.5 `int mtk::LAPACKAdapter::SolveDenseSystem ( mtk::DenseMatrix & mm, mtk::UniStgGrid2D & rhs )`  
`[static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

#### Parameters

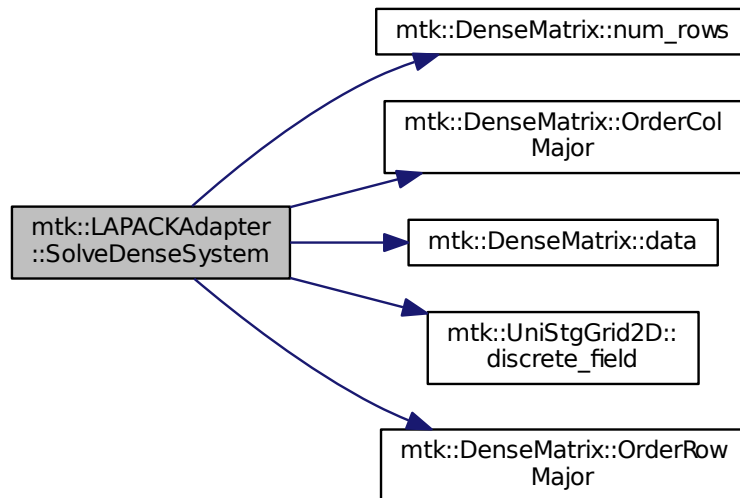
<code>in</code>	<code>matrix</code>	Input matrix.
<code>in</code>	<code>rhs</code>	Input right-hand side from info on a grid.

#### Exceptions

<code>std::bad_alloc</code>	
-----------------------------	--

Definition at line 556 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



17.16.2.6 `int mtk::LAPACKAdapter::SolveRectangularDenseSystem ( const mtk::DenseMatrix & aa, mtk::Real * ob_, int ob_id_ ) [static]`

Adapts the MTK to LAPACK's routine.

#### Parameters

<code>in, out</code>	<code>matrix</code>	Input matrix.
----------------------	---------------------	---------------

**Returns**

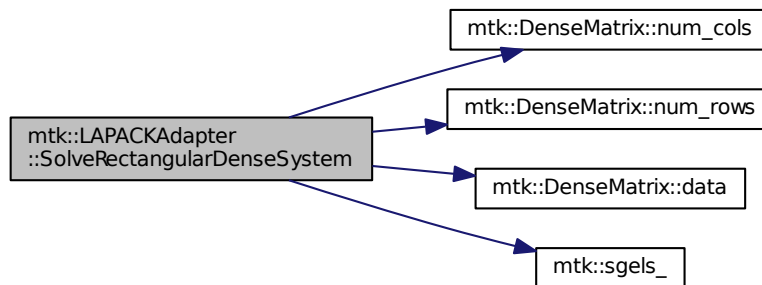
Success of the solution.

**Exceptions**

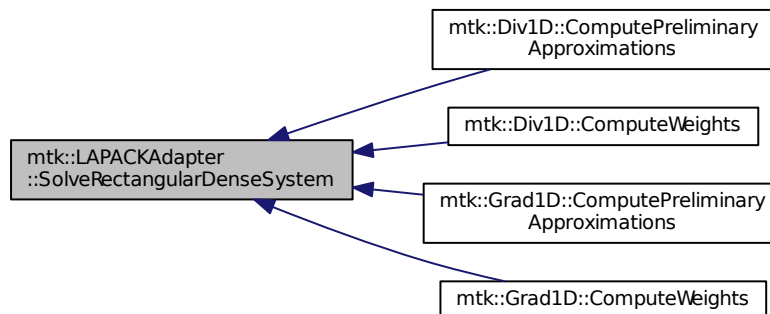
<code>std::bad_alloc</code>	
-----------------------------	--

Definition at line 791 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

- [include/mtk\\_lapack\\_adapter.h](#)
- [src/mtk\\_lapack\\_adapter.cc](#)

## 17.17 mtk::Matrix Class Reference

Definition of the representation of a matrix in the MTK.

```
#include <mtk_matrix.h>
```

Collaboration diagram for mtk::Matrix:

mtk::Matrix
<ul style="list-style-type: none"> <li>- storage_</li> <li>- ordering_</li> <li>- num_rows_</li> <li>- num_cols_</li> <li>- num_values_</li> <li>- ld_</li> <li>- num_zero_</li> <li>- num_non_zero_</li> <li>- num_null_</li> <li>- num_non_null_</li> <li>and 7 more...</li> </ul>
<ul style="list-style-type: none"> <li>+ Matrix()</li> <li>+ Matrix()</li> <li>+ ~Matrix()</li> <li>+ storage()</li> <li>+ ordering()</li> <li>+ num_rows()</li> <li>+ num_cols()</li> <li>+ num_values()</li> <li>+ ld()</li> <li>+ num_zero()</li> <li>and 18 more...</li> </ul>

## Public Member Functions

- [Matrix](#) ()  
*Default constructor.*
- [Matrix](#) (const [Matrix](#) &in)  
*Copy constructor.*
- [~Matrix](#) () noexcept  
*Destructor.*
- [MatrixStorage](#) storage () const noexcept  
*Gets the type of storage of this matrix.*
- [MatrixOrdering](#) ordering () const noexcept  
*Gets the type of ordering of this matrix.*
- int [num\\_rows](#) () const noexcept  
*Gets the number of rows.*
- int [num\\_cols](#) () const noexcept  
*Gets the number of rows.*

- int [num\\_values](#) () const noexcept  
*Gets the number of values.*
- int [ld](#) () const noexcept  
*Gets the matrix' leading dimension.*
- int [num\\_zero](#) () const noexcept  
*Gets the number of zeros.*
- int [num\\_non\\_zero](#) () const noexcept  
*Gets the number of non-zero values.*
- int [num\\_null](#) () const noexcept  
*Gets the number of null values.*
- int [num\\_non\\_null](#) () const noexcept  
*Gets the number of non-null values.*
- int [kl](#) () const noexcept  
*Gets the number of lower diagonals.*
- int [ku](#) () const noexcept  
*Gets the number of upper diagonals.*
- int [bandwidth](#) () const noexcept  
*Gets the bandwidth.*
- [Real abs\\_density](#) () const noexcept  
*Gets the absolute density.*
- [Real rel\\_density](#) () const noexcept  
*Gets the relative density.*
- [Real abs\\_sparsity](#) () const noexcept  
*Gets the Absolute sparsity.*
- [Real rel\\_sparsity](#) () const noexcept  
*Gets the Relative sparsity.*
- void [set\\_storage](#) (const [MatrixStorage](#) &tt) noexcept  
*Sets the storage type of the matrix.*
- void [set\\_ordering](#) (const [MatrixOrdering](#) &oo) noexcept  
*Sets the ordering of the matrix.*
- void [set\\_num\\_rows](#) (const int &num\_rows) noexcept  
*Sets the number of rows of the matrix.*
- void [set\\_num\\_cols](#) (const int &num\_cols) noexcept  
*Sets the number of columns of the matrix.*
- void [set\\_num\\_zero](#) (const int &in) noexcept  
*Sets the number of zero values of the matrix that matter.*
- void [set\\_num\\_null](#) (const int &in) noexcept  
*Sets the number of zero values of the matrix that DO NOT matter.*
- void [IncreaseNumZero](#) () noexcept  
*Increases the number of values that equal zero but with meaning.*
- void [IncreaseNumNull](#) () noexcept  
*Increases the number of values that equal zero but with no meaning.*

## Private Attributes

- [MatrixStorage storage\\_](#)  
*What type of matrix is this?*
- [MatrixOrdering ordering\\_](#)  
*What kind of ordering is it following?*
- int [num\\_rows\\_](#)  
*Number of rows.*
- int [num\\_cols\\_](#)  
*Number of columns.*
- int [num\\_values\\_](#)  
*Number of total values in matrix.*
- int [ld\\_](#)  
*Elements between successive rows when row-major.*
- int [num\\_zero\\_](#)  
*Number of zeros.*
- int [num\\_non\\_zero\\_](#)  
*Number of non-zero values.*
- int [num\\_null\\_](#)  
*Number of null (insignificant) values.*
- int [num\\_non\\_null\\_](#)  
*Number of null (significant) values.*
- int [kl\\_](#)  
*Number of lower diagonals on a banded matrix.*
- int [ku\\_](#)  
*Number of upper diagonals on a banded matrix.*
- int [bandwidth\\_](#)  
*Bandwidth of the matrix.*
- [Real abs\\_density\\_](#)  
*Absolute density of matrix.*
- [Real rel\\_density\\_](#)  
*Relative density of matrix.*
- [Real abs\\_sparsity\\_](#)  
*Absolute sparsity of matrix.*
- [Real rel\\_sparsity\\_](#)  
*Relative sparsity of matrix.*

### 17.17.1 Detailed Description

Definition of the representation for the matrices implemented in the MTK.

Definition at line 75 of file [mtk\\_matrix.h](#).

### 17.17.2 Constructor & Destructor Documentation

#### 17.17.2.1 `mtk::Matrix::Matrix ( )`

Definition at line 67 of file [mtk\\_matrix.cc](#).



17.17.2.2 mtk::Matrix::Matrix ( const Matrix & *in* )

## Parameters

<code>in</code>	<code>in</code>	Given matrix.
-----------------	-----------------	---------------

Definition at line 86 of file [mtk\\_matrix.cc](#).

17.17.2.3 `mtk::Matrix::~~Matrix ( )` [noexcept]

Definition at line 105 of file [mtk\\_matrix.cc](#).

### 17.17.3 Member Function Documentation

17.17.3.1 `Real mtk::Matrix::abs_density ( ) const` [noexcept]

## See also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

## Returns

Absolute density of the matrix.

17.17.3.2 `mtk::Real mtk::Matrix::abs_sparsity ( ) const` [noexcept]

## See also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

## Returns

Absolute sparsity of the matrix.

Definition at line 177 of file [mtk\\_matrix.cc](#).

17.17.3.3 `int mtk::Matrix::bandwidth ( ) const` [noexcept]

## Returns

Bandwidth of the matrix.

Definition at line 167 of file [mtk\\_matrix.cc](#).

17.17.3.4 `void mtk::Matrix::IncreaseNumNull ( )` [noexcept]

**Todo** Review the definition of sparse matrices properties.

Definition at line 275 of file [mtk\\_matrix.cc](#).

17.17.3.5 void mtk::Matrix::IncreaseNumZero ( ) [noexcept]

**Todo** Review the definition of sparse matrices properties.

Definition at line 265 of file [mtk\\_matrix.cc](#).

17.17.3.6 int mtk::Matrix::kl ( ) const [noexcept]

Returns

Number of lower diagonals.

Definition at line 157 of file [mtk\\_matrix.cc](#).

17.17.3.7 int mtk::Matrix::ku ( ) const [noexcept]

Returns

Number of upper diagonals.

Definition at line 162 of file [mtk\\_matrix.cc](#).

17.17.3.8 int mtk::Matrix::ld ( ) const [noexcept]

Leading dimension of the data array is the number of elements between successive rows (for row major storage) in memory. Most of the cases, the leading dimension is the same as the number of columns.

Returns

Leading dimension of the matrix.

Definition at line 132 of file [mtk\\_matrix.cc](#).

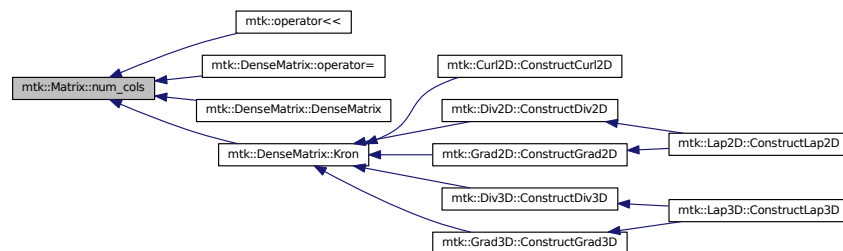
17.17.3.9 int mtk::Matrix::num\_cols ( ) const [noexcept]

Returns

Number of rows of the matrix.

Definition at line 122 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.10 `int mtk::Matrix::num_non_null ( ) const` [noexcept]

See also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

Returns

Number of non-null values of the matrix.

Definition at line 152 of file `mtk_matrix.cc`.

17.17.3.11 `int mtk::Matrix::num_non_zero ( ) const` [noexcept]

Returns

Number of non-zero values of the matrix.

Definition at line 142 of file `mtk_matrix.cc`.

17.17.3.12 `int mtk::Matrix::num_null ( ) const` [noexcept]

See also

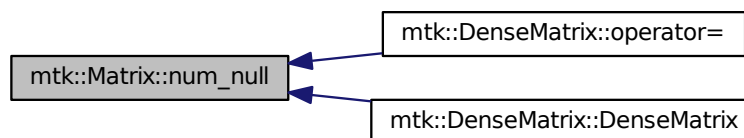
[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

Returns

Number of null values of the matrix.

Definition at line 147 of file `mtk_matrix.cc`.

Here is the caller graph for this function:



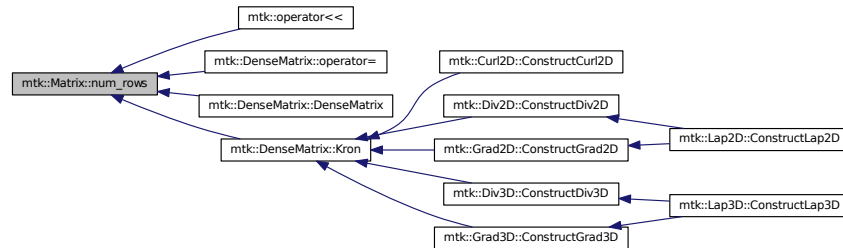
17.17.3.13 `int mtk::Matrix::num_rows ( ) const` [noexcept]

## Returns

Number of rows of the matrix.

Definition at line 117 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



### 17.17.3.14 int mtk::Matrix::num\_values ( ) const [noexcept]

## Returns

Number of values of the matrix.

Definition at line 127 of file [mtk\\_matrix.cc](#).

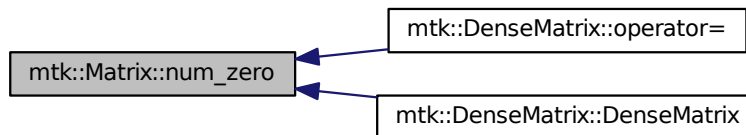
### 17.17.3.15 int mtk::Matrix::num\_zero ( ) const [noexcept]

## Returns

Number of zeros of the matrix.

Definition at line 137 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



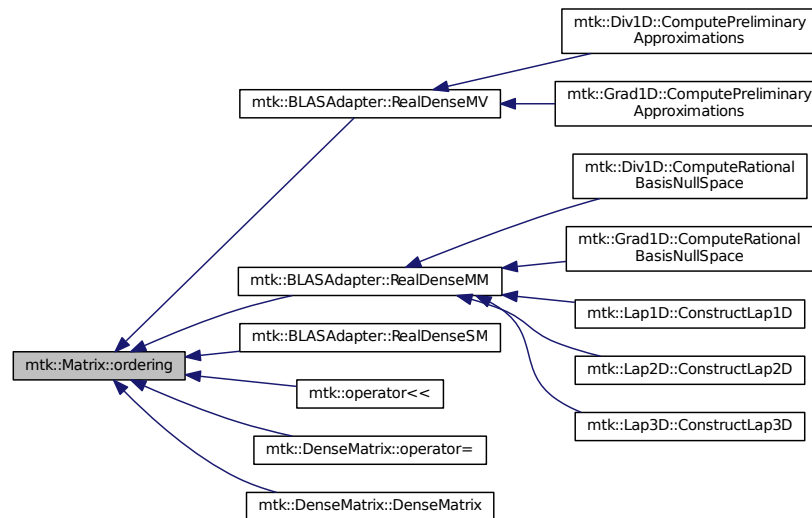
### 17.17.3.16 mtk::MatrixOrdering mtk::Matrix::ordering ( ) const [noexcept]

**Returns**

Type of ordering of this matrix.

Definition at line 112 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.17 `mtk::Real mtk::Matrix::rel_density ( ) const` [noexcept]

**See also**

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

**Returns**

Relative density of the matrix.

Definition at line 172 of file [mtk\\_matrix.cc](#).

17.17.3.18 `mtk::Real mtk::Matrix::rel_sparsity ( ) const` [noexcept]

**See also**

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

**Returns**

Relative sparsity of the matrix.

Definition at line 182 of file [mtk\\_matrix.cc](#).

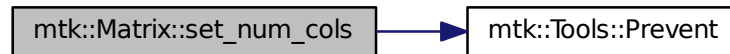
17.17.3.19 `void mtk::Matrix::set_num_cols ( const int & num_cols )` [noexcept]

## Parameters

<i>in</i>	<i>num_cols</i>	Number of columns.
-----------	-----------------	--------------------

Definition at line 225 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.20 `void mtk::Matrix::set_num_null ( const int & in ) [noexcept]`

## Parameters

<i>in</i>	<i>in</i>	Number of zero values.
-----------	-----------	------------------------

**Bug** -nan assigned on construction time due to `num_values_` being 0.

Definition at line 251 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.21 `void mtk::Matrix::set_num_rows ( const int & num_rows ) [noexcept]`

#### Parameters

<code>in</code>	<code>num_rows</code>	Number of rows.
-----------------	-----------------------	-----------------

Definition at line 213 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.22 `void mtk::Matrix::set_num_zero ( const int & in ) [noexcept]`

#### Parameters

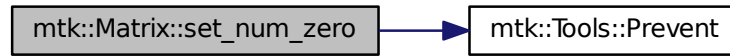
<code>in</code>	<code>in</code>	Number of zero values.
-----------------	-----------------	------------------------

**Bug** -nan assigned on construction time due to `num_values_` being 0.

Definition at line 237 of file [mtk\\_matrix.cc](#).



Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.23 `void mtk::Matrix::set_ordering ( const MatrixOrdering & oo ) [noexcept]`

See also

[MatrixOrdering](#)

#### Parameters

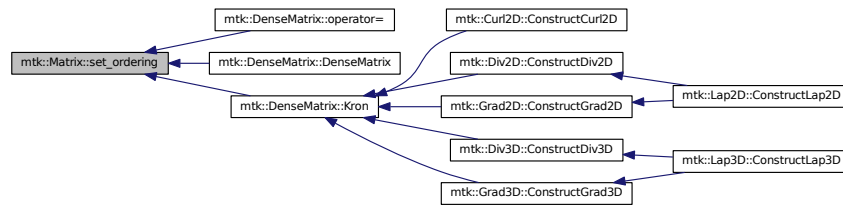
in	oo	Ordering of the matrix.
----	----	-------------------------

Definition at line 199 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.24 `void mtk::Matrix::set_storage ( const MatrixStorage & tt ) [noexcept]`

See also

[MatrixStorage](#)

Parameters

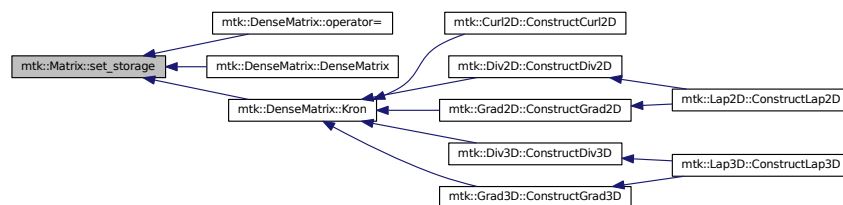
in	<i>tt</i>	Type of the matrix storage.
----	-----------	-----------------------------

Definition at line 187 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



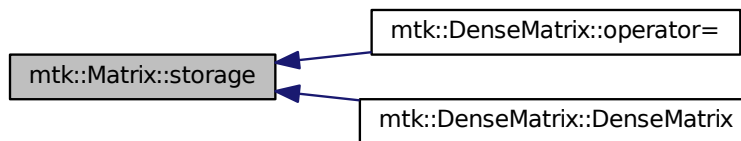
17.17.3.25 `mtk::MatrixStorage mtk::Matrix::storage ( ) const [noexcept]`

**Returns**

Type of storage of this matrix.

Definition at line 107 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:

**17.17.4 Member Data Documentation****17.17.4.1 Real mtk::Matrix::abs\_density\_ [private]**

Definition at line 296 of file [mtk\\_matrix.h](#).

**17.17.4.2 Real mtk::Matrix::abs\_sparsity\_ [private]**

Definition at line 298 of file [mtk\\_matrix.h](#).

**17.17.4.3 int mtk::Matrix::bandwidth\_ [private]**

Definition at line 294 of file [mtk\\_matrix.h](#).

**17.17.4.4 int mtk::Matrix::kl\_ [private]**

Definition at line 292 of file [mtk\\_matrix.h](#).

**17.17.4.5 int mtk::Matrix::ku\_ [private]**

Definition at line 293 of file [mtk\\_matrix.h](#).

**17.17.4.6 int mtk::Matrix::ld\_ [private]**

Definition at line 285 of file [mtk\\_matrix.h](#).

**17.17.4.7 int mtk::Matrix::num\_cols\_ [private]**

Definition at line 283 of file [mtk\\_matrix.h](#).

17.17.4.8 `int mtk::Matrix::num_non_null_ [private]`

Definition at line 290 of file [mtk\\_matrix.h](#).

17.17.4.9 `int mtk::Matrix::num_non_zero_ [private]`

Definition at line 288 of file [mtk\\_matrix.h](#).

17.17.4.10 `int mtk::Matrix::num_null_ [private]`

Definition at line 289 of file [mtk\\_matrix.h](#).

17.17.4.11 `int mtk::Matrix::num_rows_ [private]`

Definition at line 282 of file [mtk\\_matrix.h](#).

17.17.4.12 `int mtk::Matrix::num_values_ [private]`

Definition at line 284 of file [mtk\\_matrix.h](#).

17.17.4.13 `int mtk::Matrix::num_zero_ [private]`

Definition at line 287 of file [mtk\\_matrix.h](#).

17.17.4.14 **MatrixOrdering** `mtk::Matrix::ordering_ [private]`

Definition at line 280 of file [mtk\\_matrix.h](#).

17.17.4.15 **Real** `mtk::Matrix::rel_density_ [private]`

Definition at line 297 of file [mtk\\_matrix.h](#).

17.17.4.16 **Real** `mtk::Matrix::rel_sparsity_ [private]`

Definition at line 299 of file [mtk\\_matrix.h](#).

17.17.4.17 **MatrixStorage** `mtk::Matrix::storage_ [private]`

Definition at line 278 of file [mtk\\_matrix.h](#).

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

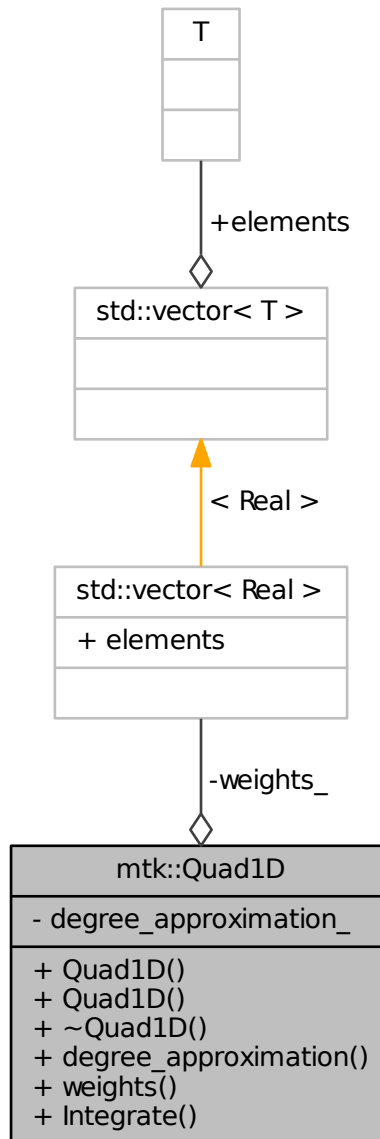
- [include/mtk\\_matrix.h](#)
- [src/mtk\\_matrix.cc](#)

## 17.18 mtk::Quad1D Class Reference

Implements a 1D mimetic quadrature.

```
#include <mtk_quad_1d.h>
```

Collaboration diagram for mtk::Quad1D:



## Public Member Functions

- [Quad1D](#) ()  
*Default constructor.*
- [Quad1D](#) (const [Quad1D](#) &quad)  
*Copy constructor.*
- [~Quad1D](#) ()  
*Destructor.*
- int [degree\\_approximation](#) () const  
*Get the degree of interpolating polynomial per sub-interval of domain.*
- [Real](#) \* [weights](#) () const  
*Return collection of weights.*
- [Real](#) [Integrate](#) ([Real](#)(\*Integrand)([Real](#) xx), [UniStgGrid1D](#) grid) const  
*Mimetic integration routine.*

## Private Attributes

- int [degree\\_approximation\\_](#)  
*Degree of the interpolating polynomial.*
- std::vector< [Real](#) > [weights\\_](#)  
*Collection of weights.*

## Friends

- std::ostream & [operator<<](#) (std::ostream &stream, [Quad1D](#) &in)  
*Output stream operator for printing.*

### 17.18.1 Detailed Description

This class implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

Definition at line 81 of file [mtk\\_quad\\_1d.h](#).

### 17.18.2 Constructor & Destructor Documentation

17.18.2.1 [mtk::Quad1D::Quad1D](#) ( )

17.18.2.2 [mtk::Quad1D::Quad1D](#) ( const [Quad1D](#) & quad )

#### Parameters

in	div	Given quadrature.
----	-----	-------------------

17.18.2.3 mtk::Quad1D::~~Quad1D ( )

### 17.18.3 Member Function Documentation

17.18.3.1 int mtk::Quad1D::degree\_approximation ( ) const

#### Returns

Degree of the interpolating polynomial per sub-interval of the domain.

17.18.3.2 Real mtk::Quad1D::Integrate ( Real(\*) (Real xx) *Integrand*, UniStgGrid1D *grid* ) const

#### Parameters

in	<i>Integrand</i>	Real-valued function to integrate.
in	<i>grid</i>	Given integration domain.

#### Returns

Result of the integration.

17.18.3.3 Real\* mtk::Quad1D::weights ( ) const

#### Returns

Collection of weights.

### 17.18.4 Friends And Related Function Documentation

17.18.4.1 std::ostream& operator<< ( std::ostream & *stream*, Quad1D & *in* ) [friend]

### 17.18.5 Member Data Documentation

17.18.5.1 int mtk::Quad1D::degree\_approximation\_ [private]

Definition at line 124 of file [mtk\\_quad\\_1d.h](#).

17.18.5.2 std::vector<Real> mtk::Quad1D::weights\_ [private]

Definition at line 126 of file [mtk\\_quad\\_1d.h](#).

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

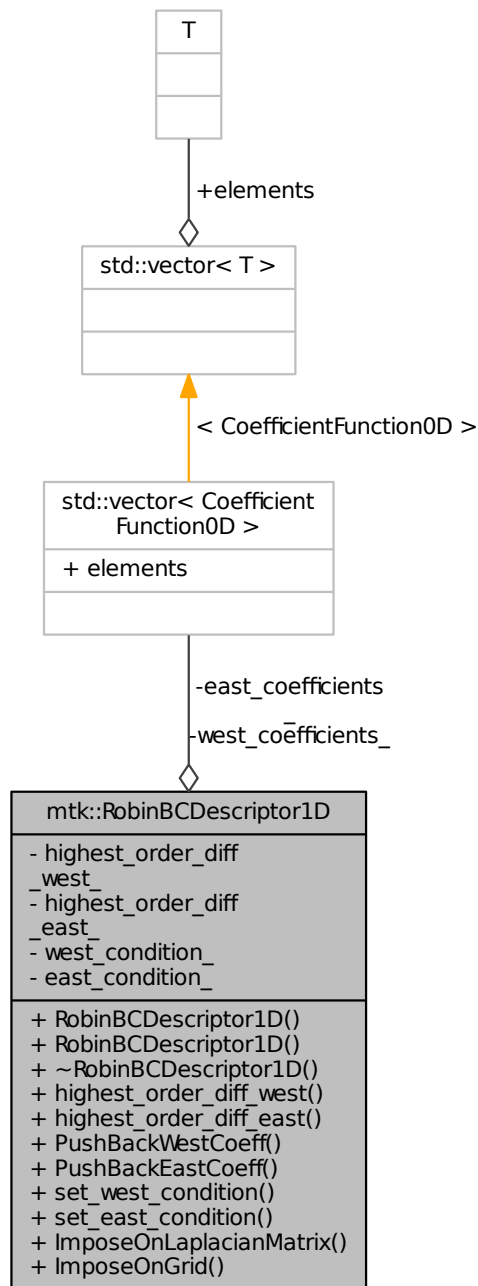
- [include/mtk\\_quad\\_1d.h](#)

## 17.19 mtk::RobinBCDescriptor1D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_1d.h>
```

Collaboration diagram for `mtk::RobinBCDescriptor1D`:



## Public Member Functions

- [RobinBCDescriptor1D \(\)](#)



*Default constructor.*

- [RobinBCDescriptor1D](#) (const [RobinBCDescriptor1D](#) &desc)

*Copy constructor.*

- [~RobinBCDescriptor1D](#) () noexcept

*Destructor.*

- int [highest\\_order\\_diff\\_west](#) () const noexcept

*Getter for the highest order of differentiation in the west boundary.*

- int [highest\\_order\\_diff\\_east](#) () const noexcept

*Getter for the highest order of differentiation in the east boundary.*

- void [PushBackWestCoeff](#) ([CoefficientFunction0D](#) cw)

*Push back coefficient function at west of lowest order diff. available.*

- void [PushBackEastCoeff](#) ([CoefficientFunction0D](#) ce)

*Push back coefficient function at east of lowest order diff. available.*

- void [set\\_west\\_condition](#) ([Real](#)(\*west\_condition)(const [Real](#) &tt)) noexcept

*Set boundary condition at west.*

- void [set\\_east\\_condition](#) ([Real](#)(\*east\_condition)(const [Real](#) &tt)) noexcept

*Set boundary condition at east.*

- bool [ImposeOnLaplacianMatrix](#) (const [Lap1D](#) &lap, [DenseMatrix](#) &matrix, const [Real](#) &time=[mtk::kZero](#)) const

*Imposes the condition on the operator represented as matrix.*

- void [ImposeOnGrid](#) ([UniStgGrid1D](#) &grid, const [Real](#) &time=[mtk::kZero](#)) const

*Imposes the condition on the grid.*

## Private Attributes

- int [highest\\_order\\_diff\\_west\\_](#)

*Highest order of differentiation for west.*

- int [highest\\_order\\_diff\\_east\\_](#)

*Highest order of differentiation for east.*

- std::vector

< [CoefficientFunction0D](#) > [west\\_coefficients\\_](#)

*Coeffs. west.*

- std::vector

< [CoefficientFunction0D](#) > [east\\_coefficients\\_](#)

*Coeffs. east.*

- [Real](#)(\* [west\\_condition\\_](#))(const [Real](#) &tt)

*Condition for west.*

- [Real](#)(\* [east\\_condition\\_](#))(const [Real](#) &tt)

*Condition for east.*

### 17.19.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 1D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context (  $\partial\Omega = \{a, b\} \subset \mathbb{R}$  ), this condition can be written as follows:

$$\begin{aligned}\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) &= \beta_a(a, t), \\ \delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) &= \beta_b(b, t).\end{aligned}$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 155 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

## 17.19.2 Constructor & Destructor Documentation

17.19.2.1 `mtk::RobinBCDescriptor1D::RobinBCDescriptor1D ( )`

Definition at line 93 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

17.19.2.2 `mtk::RobinBCDescriptor1D::RobinBCDescriptor1D ( const RobinBCDescriptor1D & desc )`

Parameters

<code>in</code>	<code>desc</code>	Given 1D descriptor.
-----------------	-------------------	----------------------

Definition at line 99 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

17.19.2.3 `mtk::RobinBCDescriptor1D::~~RobinBCDescriptor1D ( )` `[noexcept]`

Definition at line 106 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

## 17.19.3 Member Function Documentation

17.19.3.1 `int mtk::RobinBCDescriptor1D::highest_order_diff_east ( ) const` `[noexcept]`

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 113 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

17.19.3.2 `int mtk::RobinBCDescriptor1D::highest_order_diff_west ( ) const` `[noexcept]`

Returns

Integer highest order of differentiation in the west boundary.

Definition at line 108 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

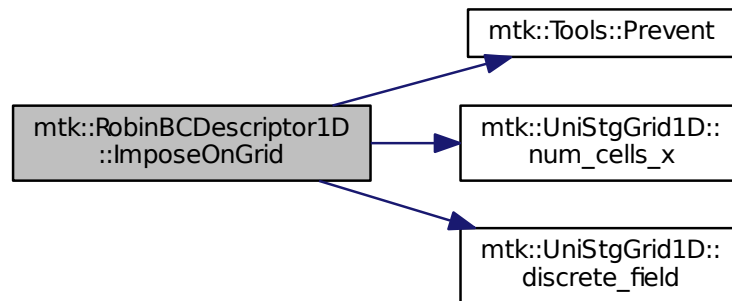
17.19.3.3 void mtk::RobinBCDescriptor1D::ImposeOnGrid ( UniStgGrid1D & *grid*, const Real & *time* = mtk::kZero ) const

#### Parameters

in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

Definition at line 246 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



17.19.3.4 bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix ( const Lap1D & *lap*, mtk::DenseMatrix & *matrix*, const Real & *time* = mtk::kZero ) const

#### Parameters

in	<i>lap</i>	Operator in the <a href="#">Matrix</a> .
in, out	<i>matrix</i>	Input Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

#### Returns

Success of the imposition.

1. Impose Dirichlet coefficients.
  - 1.1. Impose Dirichlet condition at the west.
  - 1.2. Impose Dirichlet condition at the east.
1. Impose Neumann coefficients.
  - 2.1. Create a mimetic gradient to approximate the first derivative.
  - 2.2. Extract the coefficients approximating the boundary.

**Warning**

Coefficients returned by the `mim_bndy` getter are dimensionless! Therefore we must scale them by `delta_x` (from the grid), before adding to the matrix! But this information is in the given lap!

2.3. Impose Neumann condition at the west.

2.3.1. Get gradient coefficient and scale it.

2.3.2. Multiply times the coefficient for this boundary, times the unit normal for this boundary.

2.3.3. Set the final value summing it with what is on the matrix.

2.4. Impose Neumann condition at the east.

**Warning**

The Coefficients returned by the `mim_bndy` getter are those intended for the west boundary. We must enforce the center-skew-symmetry of the resulting operator by permuting their location in the matrix, and changing their sign.

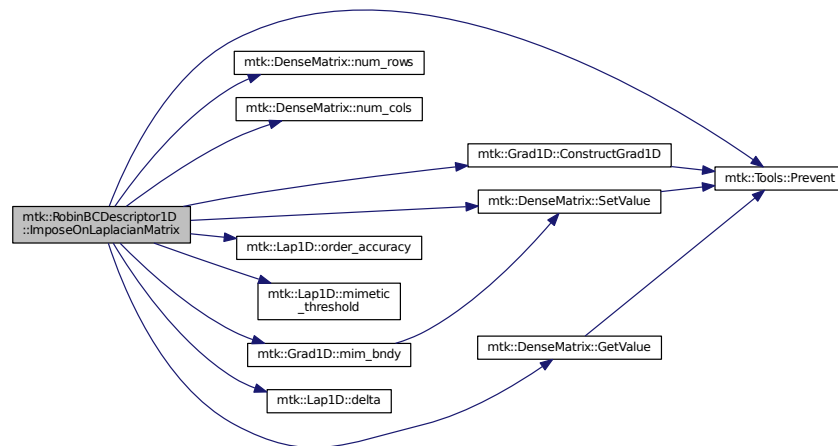
2.4.1. Get gradient coefficient and scale it.

2.4.2. Multiply times the coefficient for this boundary, times the unit normal for this boundary, and change the sign to enforce center-skew-symmetry.

2.4.3. Set the final value summing it with what is on the matrix.

Definition at line 166 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



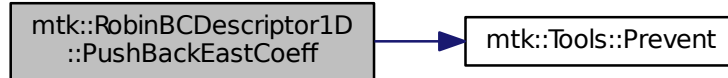
17.19.3.5 void `mtk::RobinBCDescriptor1D::PushBackEastCoeff` ( `mtk::CoefficientFunction0D ce` )

## Parameters

<i>in</i>	<i>ce</i>	Function $c_e(x, y) : \Omega \mapsto \mathbb{R}$ .
-----------	-----------	----------------------------------------------------

Definition at line 132 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



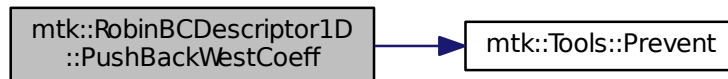
### 17.19.3.6 void mtk::RobinBCDescriptor1D::PushBackWestCoeff ( mtk::CoefficientFunction0D *cw* )

## Parameters

<i>in</i>	<i>cw</i>	Function $c_w(x, y) : \Omega \mapsto \mathbb{R}$ .
-----------	-----------	----------------------------------------------------

Definition at line 118 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



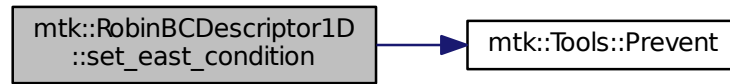
### 17.19.3.7 void mtk::RobinBCDescriptor1D::set\_east\_condition ( Real(\*) (const Real &tt) *east\_condition* ) [noexcept]

## Parameters

<i>in</i>	<i>east_condition</i>	$\beta_e(y, t) : \Omega \mapsto \mathbb{R}$ .
-----------	-----------------------	-----------------------------------------------

Definition at line 156 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



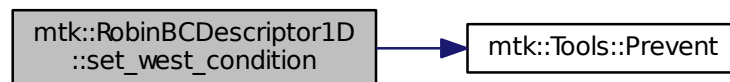
17.19.3.8 void mtk::RobinBCDescriptor1D::set\_west\_condition ( Real(\*) (const Real &tt) *west\_condition* ) [noexcept]

Parameters

in	<i>west_condition</i>	$\beta_w(y, t) : \Omega \mapsto \mathbb{R}.$
----	-----------------------	----------------------------------------------

Definition at line 146 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

Here is the call graph for this function:



## 17.19.4 Member Data Documentation

17.19.4.1 std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::east\_coefficients\_ [private]

Definition at line 237 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

17.19.4.2 Real(\*) mtk::RobinBCDescriptor1D::east\_condition\_ (const Real &tt) [private]

Definition at line 240 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

17.19.4.3 int mtk::RobinBCDescriptor1D::highest\_order\_diff\_east\_ [private]

Definition at line 234 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

17.19.4.4 int mtk::RobinBCDescriptor1D::highest\_order\_diff\_west\_ [private]

Definition at line 233 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

17.19.4.5 `std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::west_coefficients_` [private]

Definition at line 236 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

17.19.4.6 `Real(* mtk::RobinBCDescriptor1D::west_condition_)(const Real &tt)` [private]

Definition at line 239 of file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

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

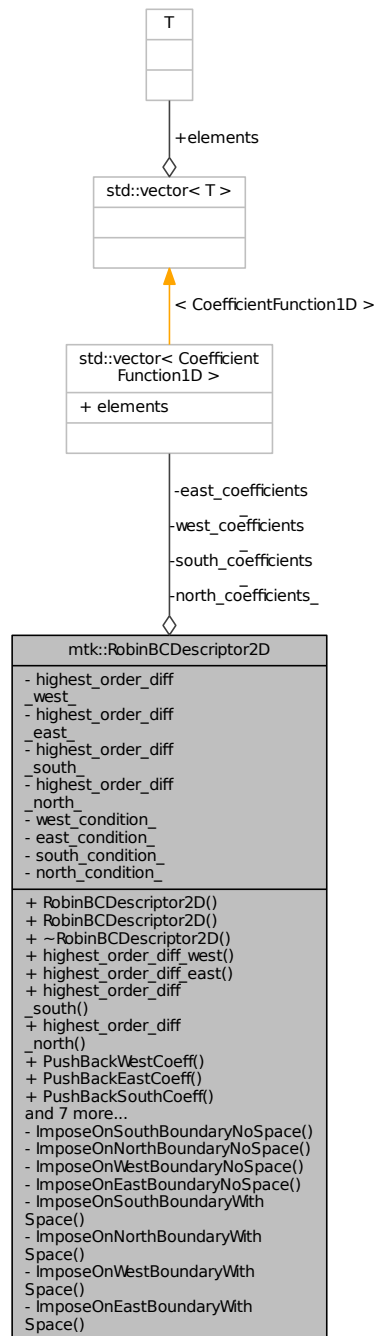
- [include/mtk\\_robin\\_bc\\_descriptor\\_1d.h](#)
- [src/mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#)

## 17.20 mtk::RobinBCDescriptor2D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_2d.h>
```

Collaboration diagram for `mtk::RobinBCDescriptor2D`:



## Public Member Functions

- [RobinBCDescriptor2D](#) ()



*Default constructor.*

- [RobinBCDescriptor2D](#) (const [RobinBCDescriptor2D](#) &desc)

*Copy constructor.*

- [~RobinBCDescriptor2D](#) () noexcept

*Destructor.*

- int [highest\\_order\\_diff\\_west](#) () const noexcept

*Getter for the highest order of differentiation in the west boundary.*

- int [highest\\_order\\_diff\\_east](#) () const noexcept

*Getter for the highest order of differentiation in the east boundary.*

- int [highest\\_order\\_diff\\_south](#) () const noexcept

*Getter for the highest order of differentiation in the south boundary.*

- int [highest\\_order\\_diff\\_north](#) () const noexcept

*Getter for the highest order of differentiation in the north boundary.*

- void [PushBackWestCoeff](#) ([CoefficientFunction1D](#) cw)

*Push back coefficient function at west of lowest order diff. available.*

- void [PushBackEastCoeff](#) ([CoefficientFunction1D](#) ce)

*Push back coefficient function at east of lowest order diff. available.*

- void [PushBackSouthCoeff](#) ([CoefficientFunction1D](#) cs)

*Push back coefficient function south of lowest order diff. available.*

- void [PushBackNorthCoeff](#) ([CoefficientFunction1D](#) cn)

*Push back coefficient function north of lowest order diff. available.*

- void [set\\_west\\_condition](#) ([Real](#)(\*west\_condition)(const [Real](#) &yy, const [Real](#) &tt)) noexcept

*Set boundary condition at west.*

- void [set\\_east\\_condition](#) ([Real](#)(\*east\_condition)(const [Real](#) &yy, const [Real](#) &tt)) noexcept

*Set boundary condition at east.*

- void [set\\_south\\_condition](#) ([Real](#)(\*south\_condition)(const [Real](#) &xx, const [Real](#) &tt)) noexcept

*Set boundary condition at south.*

- void [set\\_north\\_condition](#) ([Real](#)(\*north\_condition)(const [Real](#) &xx, const [Real](#) &tt)) noexcept

*Set boundary condition at north.*

- bool [ImposeOnLaplacianMatrix](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

*Imposes the condition on the operator represented as matrix.*

- void [ImposeOnGrid](#) ([UniStgGrid2D](#) &grid, const [Real](#) &time=kZero) const

*Imposes the condition on the grid.*

## Private Member Functions

- bool [ImposeOnSouthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

*Imposes the condition on the south boundary.*

- bool [ImposeOnNorthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

*Imposes the condition on the north boundary.*

- bool [ImposeOnWestBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

*Imposes the condition on the west boundary.*

- bool [ImposeOnEastBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

*Imposes the condition on the east boundary.*

- bool [ImposeOnSouthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the south boundary.*

- bool [ImposeOnNorthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the north boundary.*

- bool [ImposeOnWestBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the west boundary.*

- bool [ImposeOnEastBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the east boundary.*

## Private Attributes

- int [highest\\_order\\_diff\\_west\\_](#)

*Highest order of differentiation west.*

- int [highest\\_order\\_diff\\_east\\_](#)

*Highest order of differentiation east.*

- int [highest\\_order\\_diff\\_south\\_](#)

*Highest order differentiation for south.*

- int [highest\\_order\\_diff\\_north\\_](#)

*Highest order differentiation for north.*

- std::vector  
< [CoefficientFunction1D](#) > [west\\_coefficients\\_](#)  
*Coeffs. west.*

- std::vector  
< [CoefficientFunction1D](#) > [east\\_coefficients\\_](#)  
*Coeffs. east.*

- std::vector  
< [CoefficientFunction1D](#) > [south\\_coefficients\\_](#)  
*Coeffs. south.*

- std::vector  
< [CoefficientFunction1D](#) > [north\\_coefficients\\_](#)  
*Coeffs. north.*

- [Real](#)(\* [west\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &tt)  
*Condition west.*

- [Real](#)(\* [east\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &tt)  
*Condition east.*

- [Real](#)(\* [south\\_condition\\_](#))(const [Real](#) &yy, const [Real](#) &tt)  
*Cond. south.*

- [Real](#)(\* [north\\_condition\\_](#))(const [Real](#) &yy, const [Real](#) &tt)  
*Cond. north.*

### 17.20.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 2D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 132 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

### 17.20.2 Constructor & Destructor Documentation

#### 17.20.2.1 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D ( )

Definition at line 84 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

#### 17.20.2.2 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D ( const RobinBCDescriptor2D & desc )

Parameters

<i>in</i>	<i>desc</i>	Given 2D descriptor.
-----------	-------------	----------------------

Definition at line 94 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

#### 17.20.2.3 mtk::RobinBCDescriptor2D::~~RobinBCDescriptor2D ( ) [noexcept]

Definition at line 105 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

### 17.20.3 Member Function Documentation

#### 17.20.3.1 int mtk::RobinBCDescriptor2D::highest\_order\_diff\_east ( ) const [noexcept]

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 112 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

17.20.3.2 `int mtk::RobinBCDescriptor2D::highest_order_diff_north ( ) const` `[noexcept]`

#### Returns

Integer highest order of differentiation in the north boundary.

Definition at line 122 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

17.20.3.3 `int mtk::RobinBCDescriptor2D::highest_order_diff_south ( ) const` `[noexcept]`

#### Returns

Integer highest order of differentiation in the south boundary.

Definition at line 117 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

17.20.3.4 `int mtk::RobinBCDescriptor2D::highest_order_diff_west ( ) const` `[noexcept]`

#### Returns

Integer highest order of differentiation in the west boundary.

Definition at line 107 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

17.20.3.5 `bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const` `[private]`

#### Parameters

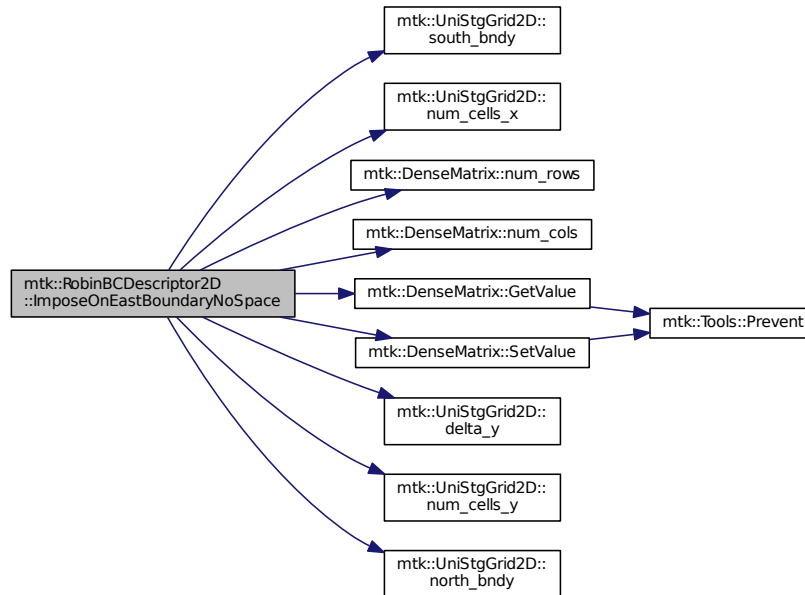
<i>in</i>	<i>lap</i>	Laplacian operator on the matrix.
<i>in</i>	<i>grid</i>	Grid upon which impose the desired boundary condition.
<i>in, out</i>	<i>matrix</i>	Input matrix with the Laplacian operator.
<i>in</i>	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.

2. Impose the Neumann condition.

Definition at line 495 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.6 `bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const [private]`

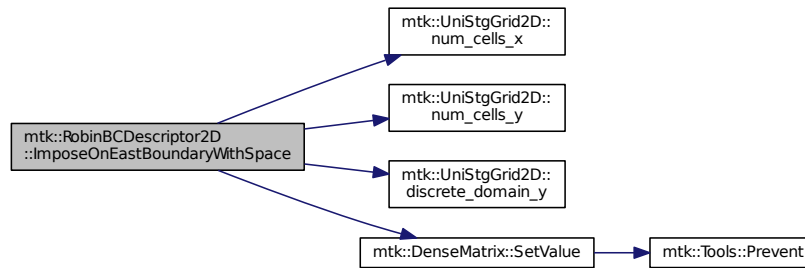
#### Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 564 of file `mtk_robin_bc_descriptor_2d.cc`.

Here is the call graph for this function:



17.20.3.7 void mtk::RobinBCDescriptor2D::ImposeOnGrid ( mtk::UniStgGrid2D & *grid*, const Real & *time* = kZero ) const

#### Parameters

in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose assuming an scalar grid.

1.1. Impose south condition.

1.1.1. Impose south-west corner.

1.1.2. Impose south border.

1.1.3. Impose south-east corner.

1.2. Impose north condition.

1.2.1. Impose north-west corner.

1.2.2. Impose north border.

1.2.3. Impose north-east corner.

1.3. Impose west condition.

1.3.1. Impose south-west corner.

#### Note

As per discussion with Otilio, we will take the **arithmetic mean** of the values of the BCs at the corners.

1.3.2. Impose west border.

1.3.3. Impose north-west corner.

1.4. Impose east condition.

1.4.1. Impose south-east corner.

1.4.2. Impose east border.

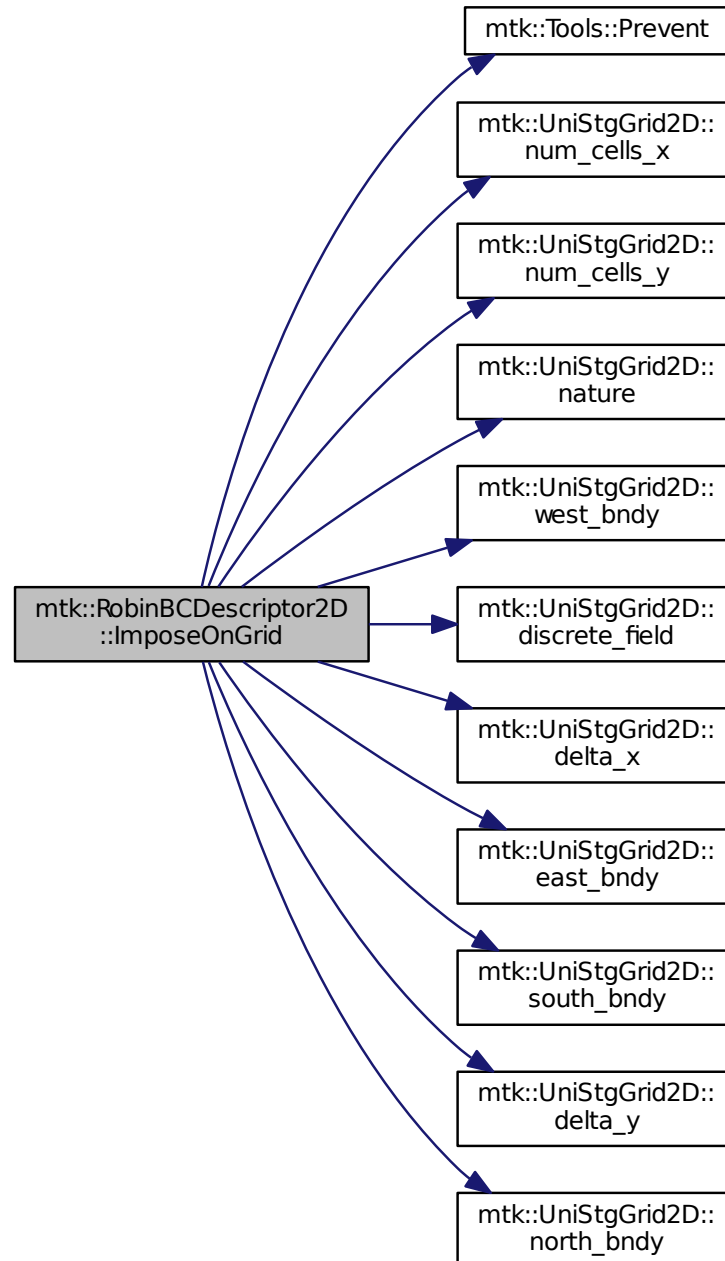
1.4.3. Impose north-east corner.

1. Impose assuming a vector grid.

**Todo** Implement imposition for vector-valued grids. Need research here!

Definition at line 674 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.8 `bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const`

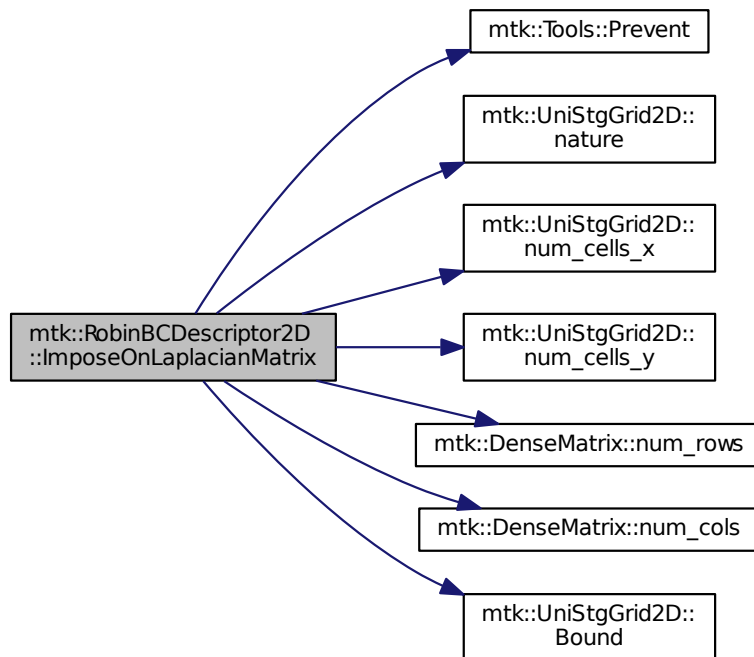
#### Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

If we have not bound anything to the grid, then we have to generate our collection of spatial coordinates, as we evaluate the coefficients.

Definition at line 591 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.9 `bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const [private]`

#### Parameters

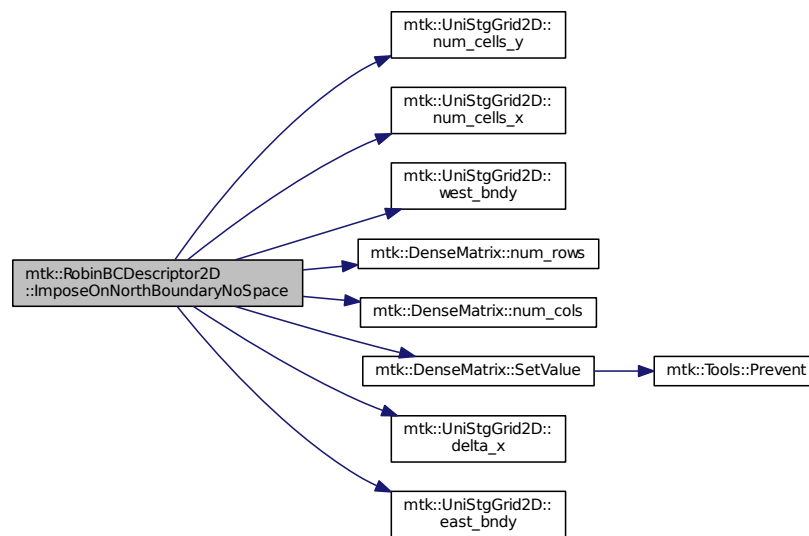


in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 312 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.10 `bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const [private]`

#### Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose Dirichlet condition.

For each entry on the diagonal:

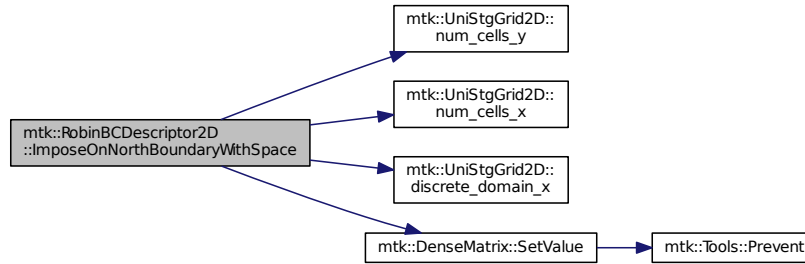
Evaluate next set spatial coordinates to evaluate the coefficient.

Evaluate and assign the Dirichlet coefficient.

1. Impose the Neumann condition.

Definition at line 372 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.11 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const [private]`

#### Parameters

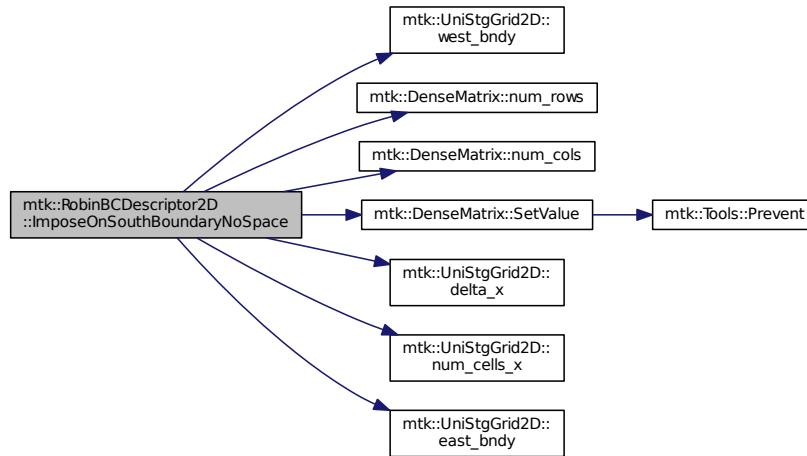
in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

**Todo** Impose the Neumann conditions on every pole, for every scenario.

Definition at line 229 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.12 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const [private]`

#### Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

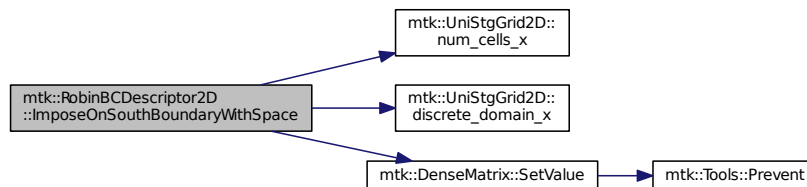
1. Impose the Dirichlet condition first.

**Todo** Impose Harmonic mean on the corners for the case when the generated space is available, for all poles.

1. Impose the Neumann condition.

Definition at line 284 of file `mtk_robin_bc_descriptor_2d.cc`.

Here is the call graph for this function:



17.20.3.13 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const` [private]

#### Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.

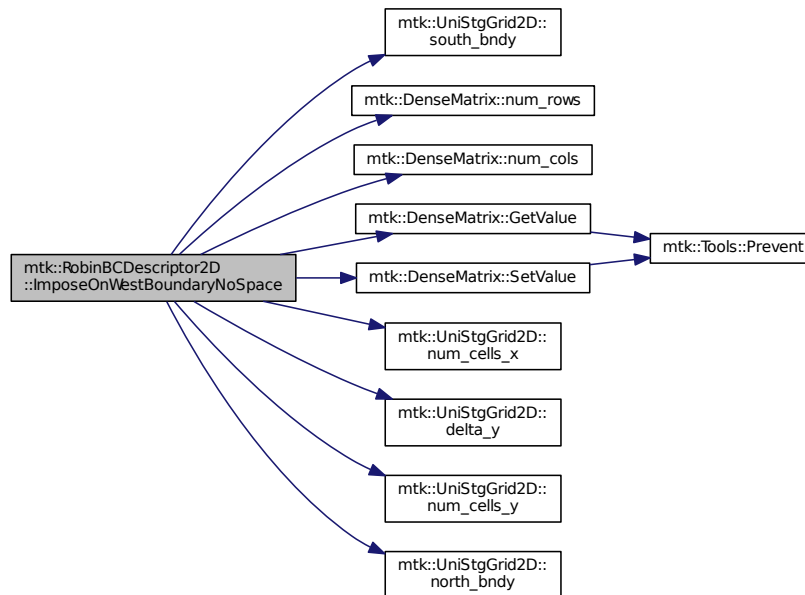
#### Note

As it can be seen, we must adopt a convention about how to treat the corners. Based on a reasoning with Otilio, we will take the **harmonic mean**.

1. Impose the Neumann condition.

Definition at line 399 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.14 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace ( const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero ) const` [private]

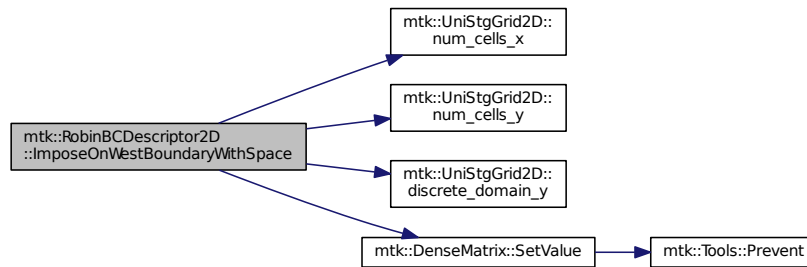
## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 468 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



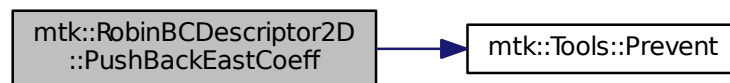
17.20.3.15 void mtk::RobinBCDescriptor2D::PushBackEastCoeff ( mtk::CoefficientFunction1D *ce* )

## Parameters

in	<i>cw</i>	Coeff. $c_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
----	-----------	----------------------------------------------------------------------------

Definition at line 141 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



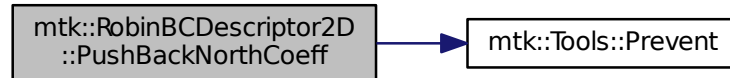
17.20.3.16 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff ( mtk::CoefficientFunction1D *cn* )

## Parameters

in	cw	Coeff. $c_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
----	----	----------------------------------------------------------------------------

Definition at line 169 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



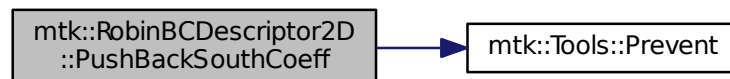
17.20.3.17 void `mtk::RobinBCDescriptor2D::PushBackSouthCoeff` ( `mtk::CoefficientFunction1D cs` )

## Parameters

in	cw	Coeff. $c_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
----	----	----------------------------------------------------------------------------

Definition at line 155 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



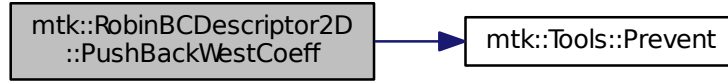
17.20.3.18 void `mtk::RobinBCDescriptor2D::PushBackWestCoeff` ( `mtk::CoefficientFunction1D cw` )

## Parameters

in	cw	Coeff. $c_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
----	----	----------------------------------------------------------------------------

Definition at line 127 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



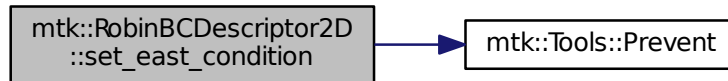
17.20.3.19 void mtk::RobinBCDescriptor2D::set\_east\_condition ( Real(\*) (const Real &yy, const Real &tt) east\_condition )  
[noexcept]

#### Parameters

in	east_condition	$\beta_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$
----	----------------	------------------------------------------------------------------------

Definition at line 194 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



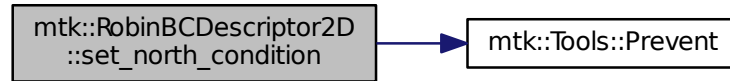
17.20.3.20 void mtk::RobinBCDescriptor2D::set\_north\_condition ( Real(\*) (const Real &xx, const Real &tt) north\_condition )  
[noexcept]

#### Parameters

in	north_condition	$\beta_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$
----	-----------------	------------------------------------------------------------------------

Definition at line 217 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



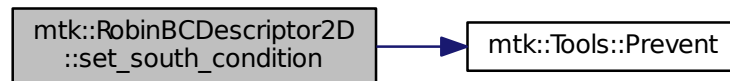
17.20.3.21 `void mtk::RobinBCDescriptor2D::set_south_condition ( Real(*) (const Real &xx, const Real &tt) south_condition )`  
`[noexcept]`

#### Parameters

<code>in</code>	<code>south_condition</code>	$\beta_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$
-----------------	------------------------------	------------------------------------------------------------------------

Definition at line 205 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

Here is the call graph for this function:



17.20.3.22 `void mtk::RobinBCDescriptor2D::set_west_condition ( Real(*) (const Real &yy, const Real &tt) west_condition )`  
`[noexcept]`

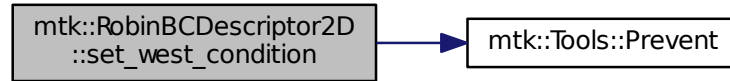
#### Parameters

<code>in</code>	<code>west_condition</code>	$\beta_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$
-----------------	-----------------------------	------------------------------------------------------------------------

Definition at line 183 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).



Here is the call graph for this function:



#### 17.20.4 Member Data Documentation

17.20.4.1 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::east_coefficients_` [private]

Definition at line 367 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.2 `Real(* mtk::RobinBCDescriptor2D::east_condition_)(const Real &xx, const Real &tt)` [private]

Definition at line 372 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.3 `int mtk::RobinBCDescriptor2D::highest_order_diff_east_` [private]

Definition at line 362 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.4 `int mtk::RobinBCDescriptor2D::highest_order_diff_north_` [private]

Definition at line 364 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.5 `int mtk::RobinBCDescriptor2D::highest_order_diff_south_` [private]

Definition at line 363 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.6 `int mtk::RobinBCDescriptor2D::highest_order_diff_west_` [private]

Definition at line 361 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.7 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::north_coefficients_` [private]

Definition at line 369 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.8 `Real(* mtk::RobinBCDescriptor2D::north_condition_)(const Real &yy, const Real &tt)` [private]

Definition at line 374 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.9 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::south_coefficients_` [private]

Definition at line 368 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.10 `Real(* mtk::RobinBCDescriptor2D::south_condition_)(const Real &yy, const Real &tt)` [private]

Definition at line 373 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.11 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::west_coefficients_` [private]

Definition at line 366 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

17.20.4.12 `Real(* mtk::RobinBCDescriptor2D::west_condition_)(const Real &xx, const Real &tt)` [private]

Definition at line 371 of file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

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

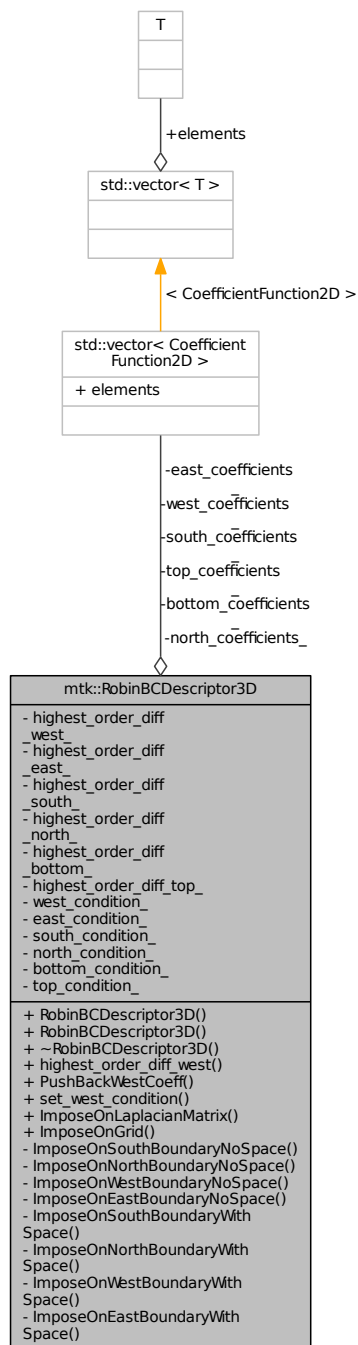
- [include/mtk\\_robin\\_bc\\_descriptor\\_2d.h](#)
- [src/mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#)

## 17.21 mtk::RobinBCDescriptor3D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_3d.h>
```

Collaboration diagram for mtk::RobinBCDescriptor3D:



## Public Member Functions

- [RobinBCDescriptor3D \(\)](#)

*Default constructor.*

- [RobinBCDescriptor3D](#) (const [RobinBCDescriptor3D](#) &desc)

*Copy constructor.*

- [~RobinBCDescriptor3D](#) () noexcept

*Destructor.*

- int [highest\\_order\\_diff\\_west](#) () const noexcept

*Getter for highest order of differentiation in the \* face.*

- void [PushBackWestCoeff](#) ([CoefficientFunction2D](#) cw)

*Push back coefficient function at west lowest order diff. available.*

- void [set\\_west\\_condition](#) ([Real](#)(\*west\_condition)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)) noexcept

*Set boundary condition at west.*

- bool [ImposeOnLaplacianMatrix](#) (const [Lap3D](#) &lap, const [UniStgGrid3D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the operator represented as matrix.*

- void [ImposeOnGrid](#) ([UniStgGrid3D](#) &grid, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the grid.*

## Private Member Functions

- bool [ImposeOnSouthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the south boundary.*

- bool [ImposeOnNorthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the north boundary.*

- bool [ImposeOnWestBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the west boundary.*

- bool [ImposeOnEastBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the east boundary.*

- bool [ImposeOnSouthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the south boundary.*

- bool [ImposeOnNorthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the north boundary.*

- bool [ImposeOnWestBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the west boundary.*

- bool [ImposeOnEastBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

*Imposes the condition on the east boundary.*

## Private Attributes

- int [highest\\_order\\_diff\\_west\\_](#)  
*Highest order of differentiation west.*
- int [highest\\_order\\_diff\\_east\\_](#)  
*Highest order of differentiation east.*
- int [highest\\_order\\_diff\\_south\\_](#)  
*Highest order differentiation for south.*
- int [highest\\_order\\_diff\\_north\\_](#)  
*Highest order differentiation for north.*
- int [highest\\_order\\_diff\\_bottom\\_](#)  
*Highest order differentiation bottom.*
- int [highest\\_order\\_diff\\_top\\_](#)  
*Highest order differentiation for top.*
- std::vector  
< [CoefficientFunction2D](#) > [west\\_coefficients\\_](#)  
*Coeffs. west.*
- std::vector  
< [CoefficientFunction2D](#) > [east\\_coefficients\\_](#)  
*Coeffs. east.*
- std::vector  
< [CoefficientFunction2D](#) > [south\\_coefficients\\_](#)  
*Coeffs. south.*
- std::vector  
< [CoefficientFunction2D](#) > [north\\_coefficients\\_](#)  
*Coeffs. north.*
- std::vector  
< [CoefficientFunction2D](#) > [bottom\\_coefficients\\_](#)  
*Coeffs. bottom.*
- std::vector  
< [CoefficientFunction2D](#) > [top\\_coefficients\\_](#)  
*Coeffs. top.*
- [Real](#)(\* [west\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Condition west.*
- [Real](#)(\* [east\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Condition east.*
- [Real](#)(\* [south\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Cond. south.*
- [Real](#)(\* [north\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Cond. north.*
- [Real](#)(\* [bottom\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Cond. bottom.*
- [Real](#)(\* [top\\_condition\\_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)  
*Cond. top.*

### 17.21.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 134 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

### 17.21.2 Constructor & Destructor Documentation

17.21.2.1 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D ( )`

17.21.2.2 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D ( const RobinBCDescriptor3D & desc )`

Parameters

<i>in</i>	<i>desc</i>	Given 2D descriptor.
-----------	-------------	----------------------

17.21.2.3 `mtk::RobinBCDescriptor3D::~~RobinBCDescriptor3D ( ) [noexcept]`

### 17.21.3 Member Function Documentation

17.21.3.1 `int mtk::RobinBCDescriptor3D::highest_order_diff_west ( ) const [noexcept]`

Returns

Integer highest order of differentiation in the \* face.

17.21.3.2 `bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryNoSpace ( const Lap2D & lap, const UniStgGrid2D & grid, DenseMatrix & matrix, const Real & time = kZero ) const [private]`

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.3 **bool** mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryWithSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.4 **void** mtk::RobinBCDescriptor3D::ImposeOnGrid ( UniStgGrid3D & *grid*, **const** Real & *time* = kZero ) **const**

## Parameters

in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.5 **bool** mtk::RobinBCDescriptor3D::ImposeOnLaplacianMatrix ( **const** Lap3D & *lap*, **const** UniStgGrid3D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const**

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.6 **bool** mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryNoSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.7 **bool** mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryWithSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.8 **bool** mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryNoSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.9 **bool** mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryWithSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.10 **bool** mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryNoSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.21.3.11 **bool** mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryWithSpace ( **const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero ) **const** [private]

## Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.



<i>in</i>	<i>time</i>	Current time snapshot. Default is kZero.
-----------	-------------	------------------------------------------

17.21.3.12 void mtk::RobinBCDescriptor3D::PushBackWestCoeff ( CoefficientFunction2D *cw* )

Parameters

<i>in</i>	<i>cw</i>	Coeff. $c_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
-----------	-----------	-------------------------------------------------------------------------------

17.21.3.13 void mtk::RobinBCDescriptor3D::set\_west\_condition ( Real(\*) (const Real &xx, const Real &yy, const Real &tt) *west\_condition* ) [noexcept]

Parameters

<i>in</i>	<i>west_condition</i>	$\beta_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$ .
-----------	-----------------------	----------------------------------------------------------------------------

## 17.21.4 Member Data Documentation

17.21.4.1 std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::bottom\_coefficients\_ [private]

Definition at line 309 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.2 Real(\* mtk::RobinBCDescriptor3D::bottom\_condition\_) (const Real &xx, const Real &yy, const Real &tt) [private]

Definition at line 324 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.3 std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::east\_coefficients\_ [private]

Definition at line 306 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.4 Real(\* mtk::RobinBCDescriptor3D::east\_condition\_) (const Real &xx, const Real &yy, const Real &tt) [private]

Definition at line 315 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.5 int mtk::RobinBCDescriptor3D::highest\_order\_diff\_bottom\_ [private]

Definition at line 302 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.6 int mtk::RobinBCDescriptor3D::highest\_order\_diff\_east\_ [private]

Definition at line 299 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.7 int mtk::RobinBCDescriptor3D::highest\_order\_diff\_north\_ [private]

Definition at line 301 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.8 `int mtk::RobinBCDescriptor3D::highest_order_diff_south_` `[private]`

Definition at line 300 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.9 `int mtk::RobinBCDescriptor3D::highest_order_diff_top_` `[private]`

Definition at line 303 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.10 `int mtk::RobinBCDescriptor3D::highest_order_diff_west_` `[private]`

Definition at line 298 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.11 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::north_coefficients_` `[private]`

Definition at line 308 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.12 `Real(* mtk::RobinBCDescriptor3D::north_condition_)(const Real &xx, const Real &yy, const Real &tt)`  
`[private]`

Definition at line 321 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.13 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::south_coefficients_` `[private]`

Definition at line 307 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.14 `Real(* mtk::RobinBCDescriptor3D::south_condition_)(const Real &xx, const Real &yy, const Real &tt)`  
`[private]`

Definition at line 318 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.15 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::top_coefficients_` `[private]`

Definition at line 310 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.16 `Real(* mtk::RobinBCDescriptor3D::top_condition_)(const Real &xx, const Real &yy, const Real &tt)` `[private]`

Definition at line 327 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.17 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::west_coefficients_` `[private]`

Definition at line 305 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

17.21.4.18 `Real(* mtk::RobinBCDescriptor3D::west_condition_)(const Real &xx, const Real &yy, const Real &tt)`  
`[private]`

Definition at line 312 of file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

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

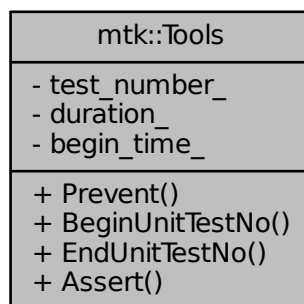
- [include/mtk\\_robin\\_bc\\_descriptor\\_3d.h](#)

## 17.22 mtk::Tools Class Reference

Tool manager class.

```
#include <mtk_tools.h>
```

Collaboration diagram for mtk::Tools:



### Static Public Member Functions

- static void [Prevent](#) (const bool complement, const char \*const fname, int lineno, const char \*const fxname) noexcept  
*Enforces preconditions by preventing their complements from occur.*
- static void [BeginUnitTestNo](#) (const int &nn) noexcept  
*Begins the execution of a unit test. Starts a timer.*
- static void [EndUnitTestNo](#) (const int &nn) noexcept  
*Ends the execution of a unit test. Stops and reports wall-clock time.*
- static void [Assert](#) (const bool &condition) noexcept  
*Asserts if the condition required to pass the unit test occurs.*

### Static Private Attributes

- static int [test\\_number\\_](#)  
*Current test being executed.*

- static [Real duration\\_](#) {}  
*Duration of the current test.*
- static clock\_t [begin\\_time\\_](#) {}  
*Elapsed time on current test.*

### 17.22.1 Detailed Description

Basic tools to ensure execution correctness, and to assists with unitary testing.

Definition at line 80 of file [mtk\\_tools.h](#).

### 17.22.2 Member Function Documentation

17.22.2.1 void [mtk::Tools::Assert](#) ( const bool & *condition* ) [static], [noexcept]

Parameters

in	<i>condition</i>	Condition to be asserted.
----	------------------	---------------------------

Definition at line 108 of file [mtk\\_tools.cc](#).

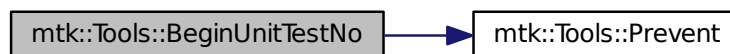
17.22.2.2 void [mtk::Tools::BeginUnitTestNo](#) ( const int & *nn* ) [static], [noexcept]

Parameters

in	<i>nn</i>	Number of the test.
----	-----------	---------------------

Definition at line 87 of file [mtk\\_tools.cc](#).

Here is the call graph for this function:



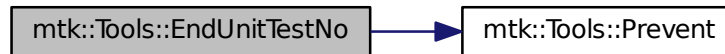
17.22.2.3 void [mtk::Tools::EndUnitTestNo](#) ( const int & *nn* ) [static], [noexcept]

Parameters

in	<i>nn</i>	Number of the test.
----	-----------	---------------------

Definition at line 99 of file [mtk\\_tools.cc](#).

Here is the call graph for this function:



**17.22.2.4** `void mtk::Tools::Prevent ( const bool complement, const char *const fname, int lineno, const char *const fxname )`  
`[static], [noexcept]`

See also

<http://stackoverflow.com/questions/8884335/print-the-file-name-line-number-and-function>

#### Parameters

in	<i>complement</i>	Complement of desired pre-condition.
in	<i>fname</i>	Name of the file being checked.
in	<i>lineno</i>	Number of the line where the check is executed.
in	<i>fxname</i>	Name of the module containing the check.

**Todo** Check if this is the best way of stalling execution.

Definition at line 62 of file [mtk\\_tools.cc](#).

### 17.22.3 Member Data Documentation

**17.22.3.1** `clock_t mtk::Tools::begin_time_ {}` `[static], [private]`

Definition at line 123 of file [mtk\\_tools.h](#).

**17.22.3.2** `mtk::Real mtk::Tools::duration_ {}` `[static], [private]`

Definition at line 121 of file [mtk\\_tools.h](#).

**17.22.3.3** `int mtk::Tools::test_number_` `[static], [private]`

Definition at line 119 of file [mtk\\_tools.h](#).

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

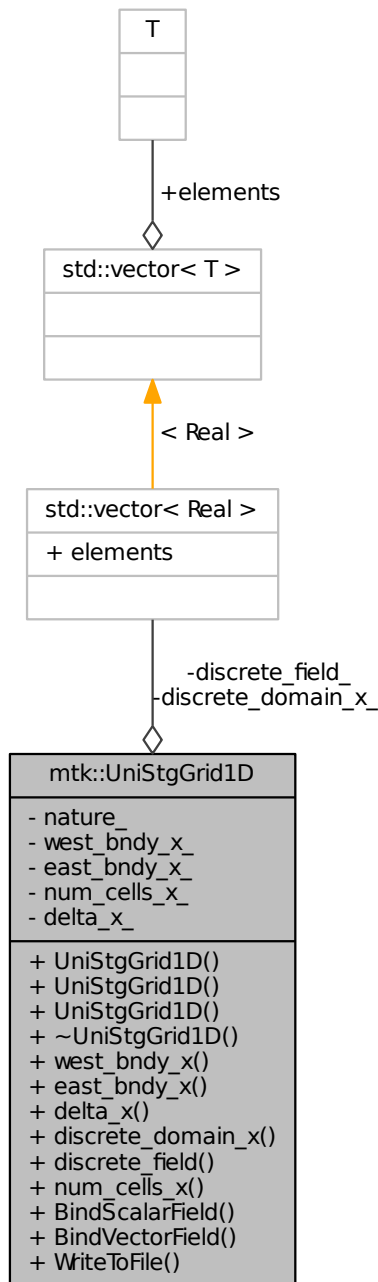
- [include/mtk\\_tools.h](#)
- [src/mtk\\_tools.cc](#)

## 17.23 mtk::UniStgGrid1D Class Reference

Uniform 1D Staggered Grid.

```
#include <mtk_uni_stg_grid_1d.h>
```

Collaboration diagram for mtk::UniStgGrid1D:



## Public Member Functions

- [UniStgGrid1D](#) ()  
*Default constructor.*
- [UniStgGrid1D](#) (const [UniStgGrid1D](#) &grid)  
*Copy constructor.*
- [UniStgGrid1D](#) (const [Real](#) &west\_bndy\_x, const [Real](#) &east\_bndy\_x, const int &num\_cells\_x, const [mtk::Field](#) &nature=mtk::SCALAR)  
*Construct a grid based on spatial discretization parameters.*
- [~UniStgGrid1D](#) ()  
*Destructor.*
- [Real west\\_bndy\\_x](#) () const  
*Provides access to west boundary spatial coordinate.*
- [Real east\\_bndy\\_x](#) () const  
*Provides access to east boundary spatial coordinate.*
- [Real delta\\_x](#) () const  
*Provides access to the computed  $\Delta x$ .*
- const [Real](#) \* [discrete\\_domain\\_x](#) () const  
*Provides access to the grid spatial data.*
- [Real](#) \* [discrete\\_field](#) ()  
*Provides access to the grid field data.*
- int [num\\_cells\\_x](#) () const  
*Provides access to the number of cells of the grid.*
- void [BindScalarField](#) ([Real](#)(\*ScalarField)(const [Real](#) &xx))  
*Binds a given scalar field to the grid.*
- void [BindVectorField](#) ([Real](#)(\*VectorField)([Real](#) xx))  
*Binds a given vector field to the grid.*
- bool [WriteToFile](#) (std::string filename, std::string space\_name, std::string field\_name) const  
*Writes grid to a file compatible with gnuplot 4.6.*

## Private Attributes

- [FieldNature nature\\_](#)  
*Nature of the discrete field.*
- std::vector< [Real](#) > [discrete\\_domain\\_x\\_](#)  
*Array of spatial data.*
- std::vector< [Real](#) > [discrete\\_field\\_](#)  
*Array of field's data.*
- [Real west\\_bndy\\_x\\_](#)  
*West boundary spatial coordinate.*
- [Real east\\_bndy\\_x\\_](#)  
*East boundary spatial coordinate.*
- [Real num\\_cells\\_x\\_](#)  
*Number of cells discretizing the domain.*
- [Real delta\\_x\\_](#)  
*Produced  $\Delta x$ .*

## Friends

- `std::ostream & operator<< (std::ostream &stream, UniStgGrid1D &in)`  
*Prints the grid as a tuple of arrays.*

### 17.23.1 Detailed Description

Uniform 1D Staggered Grid.

Definition at line 77 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

### 17.23.2 Constructor & Destructor Documentation

#### 17.23.2.1 `mtk::UniStgGrid1D::UniStgGrid1D ( )`

Definition at line 99 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

#### 17.23.2.2 `mtk::UniStgGrid1D::UniStgGrid1D ( const UniStgGrid1D &grid )`

##### Parameters

<code>in</code>	<code>grid</code>	Given grid.
-----------------	-------------------	-------------

Definition at line 108 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

#### 17.23.2.3 `mtk::UniStgGrid1D::UniStgGrid1D ( const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const mtk::FieldNature &nature = mtk::SCALAR )`

##### Parameters

<code>in</code>	<code>west_bndy_x</code>	Coordinate for the west boundary.
<code>in</code>	<code>east_bndy_x</code>	Coordinate for the east boundary.
<code>in</code>	<code>num_cells_x</code>	Number of cells of the required grid.
<code>in</code>	<code>nature</code>	Nature of the discrete field to hold.

##### See also

[mtk::FieldNature](#)

Definition at line 124 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the call graph for this function:





## 17.23.2.4 mtk::UniStgGrid1D::~~UniStgGrid1D ( )

Definition at line 144 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

## 17.23.3 Member Function Documentation

17.23.3.1 void mtk::UniStgGrid1D::BindScalarField ( *Real*(\*) (const *Real* &xx) *ScalarField* )

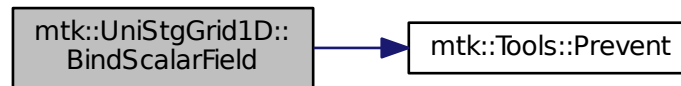
## Parameters

<i>in</i>	<i>ScalarField</i>	Pointer to the function implementing the scalar field.
-----------	--------------------	--------------------------------------------------------

1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 176 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the call graph for this function:

17.23.3.2 void mtk::UniStgGrid1D::BindVectorField ( *Real*(\*)(*Real* xx) *VectorField* )

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = v(x)\hat{\mathbf{i}}$$

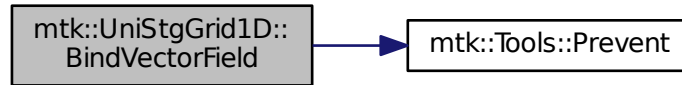
## Parameters

<i>in</i>	<i>VectorField</i>	Pointer to the function implementing the vector field.
-----------	--------------------	--------------------------------------------------------

1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 212 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the call graph for this function:



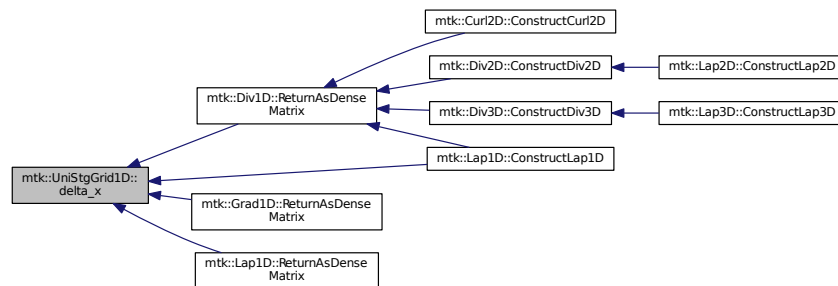
### 17.23.3.3 `mtk::Real mtk::UniStgGrid1D::delta_x ( ) const`

#### Returns

Computed  $\Delta x$ .

Definition at line 156 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the caller graph for this function:



### 17.23.3.4 `const mtk::Real * mtk::UniStgGrid1D::discrete_domain_x ( ) const`

#### Returns

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 161 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

### 17.23.3.5 `mtk::Real * mtk::UniStgGrid1D::discrete_field ( )`

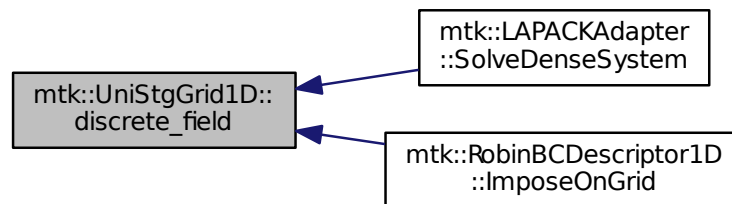
**Returns**

Pointer to the field data.

**Todo** Review const-correctness of the pointer we return. Look at the STL!

Definition at line 166 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the caller graph for this function:

**17.23.3.6 mtk::Real mtk::UniStgGrid1D::east\_bndy\_x ( ) const****Returns**

East boundary spatial coordinate.

Definition at line 151 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

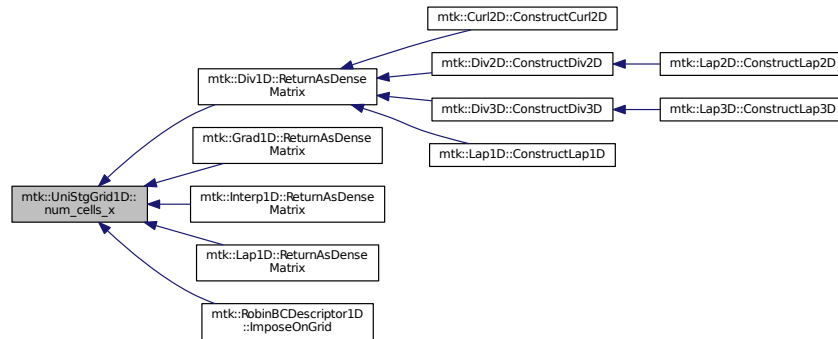
**17.23.3.7 int mtk::UniStgGrid1D::num\_cells\_x ( ) const**

**Returns**

Number of cells of the grid.

Definition at line 171 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the caller graph for this function:

**17.23.3.8 mtk::Real mtk::UniStgGrid1D::west\_bndy\_x ( ) const****Returns**

West boundary spatial coordinate.

Definition at line 146 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

**17.23.3.9 bool mtk::UniStgGrid1D::WriteToFile ( std::string filename, std::string space\_name, std::string field\_name ) const****Parameters**

in	<i>filename</i>	Name of the output file.
in	<i>space_name</i>	Name for the first column of the data.
in	<i>field_name</i>	Name for the second column of the data.

**Returns**

Success of the file writing process.

**See also**

<http://www.gnuplot.info/>

Definition at line 240 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

**17.23.4 Friends And Related Function Documentation**

17.23.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::UniStgGrid1D & in )` [friend]

1. Print spatial coordinates.
2. Print scalar field.

Definition at line 68 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

### 17.23.5 Member Data Documentation

17.23.5.1 **Real** `mtk::UniStgGrid1D::delta_x_` [private]

Definition at line 199 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.2 `std::vector<Real>` `mtk::UniStgGrid1D::discrete_domain_x_` [private]

Definition at line 193 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.3 `std::vector<Real>` `mtk::UniStgGrid1D::discrete_field_` [private]

Definition at line 194 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.4 **Real** `mtk::UniStgGrid1D::east_bndy_x_` [private]

Definition at line 197 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.5 **FieldNature** `mtk::UniStgGrid1D::nature_` [private]

Definition at line 191 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.6 **Real** `mtk::UniStgGrid1D::num_cells_x_` [private]

Definition at line 198 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

17.23.5.7 **Real** `mtk::UniStgGrid1D::west_bndy_x_` [private]

Definition at line 196 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

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

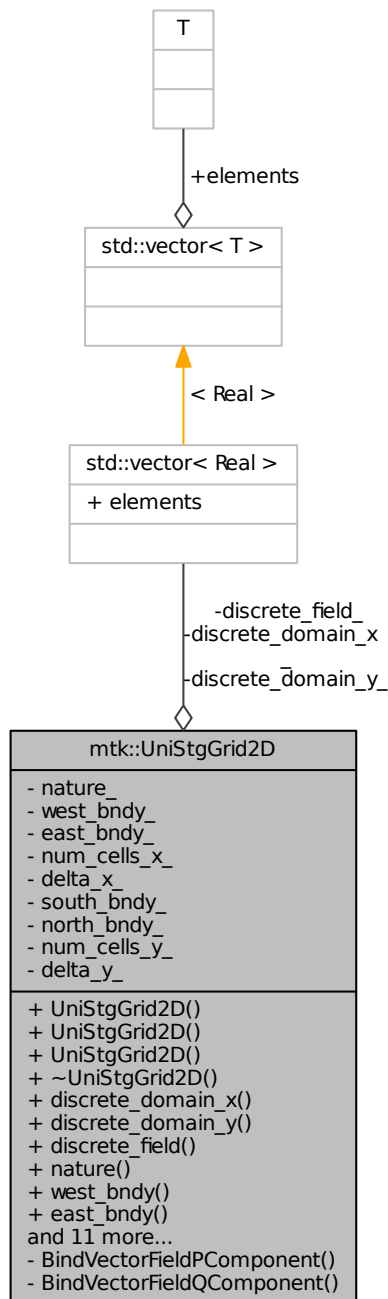
- [include/mtk\\_uni\\_stg\\_grid\\_1d.h](#)
- [src/mtk\\_uni\\_stg\\_grid\\_1d.cc](#)

## 17.24 mtk::UniStgGrid2D Class Reference

Uniform 2D Staggered Grid.

```
#include <mtk_uni_stg_grid_2d.h>
```

Collaboration diagram for `mtk::UniStgGrid2D`:



## Public Member Functions

- [UniStgGrid2D](#) ()

*Default constructor.*

- **UniStgGrid2D** (const **UniStgGrid2D** &grid)

*Copy constructor.*

- **UniStgGrid2D** (const **Real** &west\_bndy\_x, const **Real** &east\_bndy\_x, const int &num\_cells\_x, const **Real** &south\_bndy\_y, const **Real** &north\_bndy\_y, const int &num\_cells\_y, const **mtk::FieldNature** &nature=**mtk::S↔**  
**CALAR**)

*Construct a grid based on spatial discretization parameters.*

- **~UniStgGrid2D** ()

*Destructor.*

- const **Real** \* **discrete\_domain\_x** () const

*Provides access to the grid spatial data.*

- const **Real** \* **discrete\_domain\_y** () const

*Provides access to the grid spatial data.*

- **Real** \* **discrete\_field** ()

*Provides access to the grid field data.*

- **FieldNature** **nature** () const

*Physical nature of the data bound to the grid.*

- **Real** **west\_bndy** () const

*Provides access to west boundary spatial coordinate.*

- **Real** **east\_bndy** () const

*Provides access to east boundary spatial coordinate.*

- int **num\_cells\_x** () const

*Provides access to the number of cells of the grid.*

- **Real** **delta\_x** () const

*Provides access to the computed  $\Delta x$ .*

- **Real** **south\_bndy** () const

*Provides access to south boundary spatial coordinate.*

- **Real** **north\_bndy** () const

*Provides access to north boundary spatial coordinate.*

- int **num\_cells\_y** () const

*Provides access to the number of cells of the grid.*

- **Real** **delta\_y** () const

*Provides access to the computed  $\Delta y$ .*

- bool **Bound** () const

*Have any field been bound to the grid?*

- int **Size** () const

*Total number of samples in the grid.*

- void **BindScalarField** (**Real**(\*ScalarField)(const **Real** &xx, const **Real** &yy))

*Binds a given scalar field to the grid.*

- void **BindVectorField** (**Real**(\*VectorFieldPComponent)(const **Real** &xx, const **Real** &yy), **Real**(\*VectorFieldQ↔  
Component)(const **Real** &xx, const **Real** &yy))

*Binds a given vector field to the grid.*

- bool **WriteToFile** (std::string filename, std::string space\_name\_x, std::string space\_name\_y, std::string field\_↔  
name) const

*Writes grid to a file compatible with Gnuplot 4.6.*

## Private Member Functions

- void `BindVectorFieldPComponent` (`Real`(\*VectorFieldPComponent)(const `Real` &xx, const `Real` &yy))  
*Binds a given component of a vector field to the grid.*
- void `BindVectorFieldQComponent` (`Real`(\*VectorFieldQComponent)(const `Real` &xx, const `Real` &yy))  
*Binds a given component of a vector field to the grid.*

## Private Attributes

- `std::vector< Real > discrete_domain_x_`  
*Array of spatial data.*
- `std::vector< Real > discrete_domain_y_`  
*Array of spatial data.*
- `std::vector< Real > discrete_field_`  
*Array of field's data.*
- `FieldNature nature_`  
*Nature of the discrete field.*
- `Real west_bndy_`  
*West boundary spatial coordinate.*
- `Real east_bndy_`  
*East boundary spatial coordinate.*
- `int num_cells_x_`  
*Number of cells discretizing the domain.*
- `Real delta_x_`  
*Computed  $\Delta x$ .*
- `Real south_bndy_`  
*West boundary spatial coordinate.*
- `Real north_bndy_`  
*East boundary spatial coordinate.*
- `int num_cells_y_`  
*Number of cells discretizing the domain.*
- `Real delta_y_`  
*Computed  $\Delta y$ .*

## Friends

- `std::ostream & operator<<` (`std::ostream` &stream, `UniStgGrid2D` &in)  
*Prints the grid as a tuple of arrays.*

## 17.24.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 79 of file `mtk_uni_stg_grid_2d.h`.



## 17.24.2 Constructor & Destructor Documentation

### 17.24.2.1 mtk::UniStgGrid2D::UniStgGrid2D ( )

Definition at line 131 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

### 17.24.2.2 mtk::UniStgGrid2D::UniStgGrid2D ( const UniStgGrid2D & grid )

Parameters

in	<i>grid</i>	Given grid.
----	-------------	-------------

Definition at line 145 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

### 17.24.2.3 mtk::UniStgGrid2D::UniStgGrid2D ( const Real & west\_bndy\_x, const Real & east\_bndy\_x, const int & num\_cells\_x, const Real & south\_bndy\_y, const Real & north\_bndy\_y, const int & num\_cells\_y, const mtk::FieldNature & nature = mtk::SCALAR )

Parameters

in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</i>	Number of cells of the required grid.
in	<i>nature</i>	Nature of the discrete field to hold.

See also

[mtk::FieldNature](#)

Definition at line 169 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the call graph for this function:



### 17.24.2.4 mtk::UniStgGrid2D::~~UniStgGrid2D ( )

Definition at line 203 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

### 17.24.3 Member Function Documentation

17.24.3.1 void mtk::UniStgGrid2D::BindScalarField ( Real(\*) (const Real &xx, const Real &yy) *ScalarField* )

#### Parameters

in	<i>ScalarField</i>	Pointer to the function implementing the scalar field.
----	--------------------	--------------------------------------------------------

1. Create collection of spatial coordinates for  $x$ .
2. Create collection of spatial coordinates for  $y$ .
3. Create collection of field samples.

Definition at line 275 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the call graph for this function:



17.24.3.2 void mtk::UniStgGrid2D::BindVectorField ( Real(\*) (const Real &xx, const Real &yy) *VectorFieldPComponent*, Real(\*) (const Real &xx, const Real &yy) *VectorFieldQComponent* )

We assume the field to be of the form:

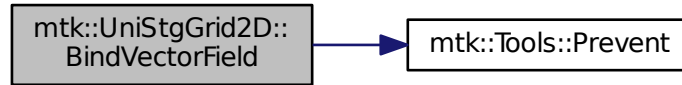
$$\mathbf{v}(\mathbf{x}) = p(x, y) \hat{\mathbf{i}} + q(x, y) \hat{\mathbf{j}}$$

#### Parameters

in	<i>VectorFieldPComponent</i>	Pointer to the function implementing the $p$ component of the vector field.
in	<i>VectorFieldQComponent</i>	Pointer to the function implementing the $q$ component of the vector field.

Definition at line 423 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the call graph for this function:



17.24.3.3 void mtk::UniStgGrid2D::BindVectorFieldPComponent ( Real(\*) (const Real &xx, const Real &yy) VectorFieldPComponent ) [private]

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = p(x, y)\hat{\mathbf{i}} + q(x, y)\hat{\mathbf{j}}$$

Parameters

in	<i>BindVectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
----	----------------------------------	---------------------------------------------------------------------------------

1. Create collection of spatial coordinates for  $x$ .
2. Create collection of spatial coordinates for  $y$ .
3. Allocate space for discrete vector field and bind \$ p \$ component.

Definition at line 330 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

17.24.3.4 void mtk::UniStgGrid2D::BindVectorFieldQComponent ( Real(\*) (const Real &xx, const Real &yy) VectorFieldQComponent ) [private]

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = p(x, y)\hat{\mathbf{i}} + q(x, y)\hat{\mathbf{j}}$$

Parameters

in	<i>BindVectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
----	----------------------------------	---------------------------------------------------------------------------------

1. Bind \$ q \$ component, since \$ p \$ component has already been bound.

Definition at line 395 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

17.24.3.5 bool mtk::UniStgGrid2D::Bound ( ) const

**Returns**

True is a field has been bound.

Definition at line 255 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

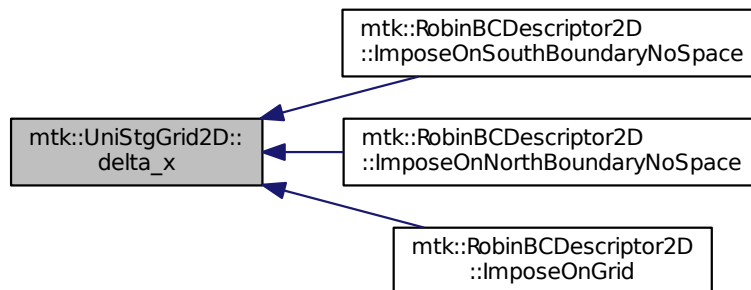
Here is the caller graph for this function:

**17.24.3.6 mtk::Real mtk::UniStgGrid2D::delta\_x ( ) const****Returns**

Computed  $\Delta x$ .

Definition at line 225 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:

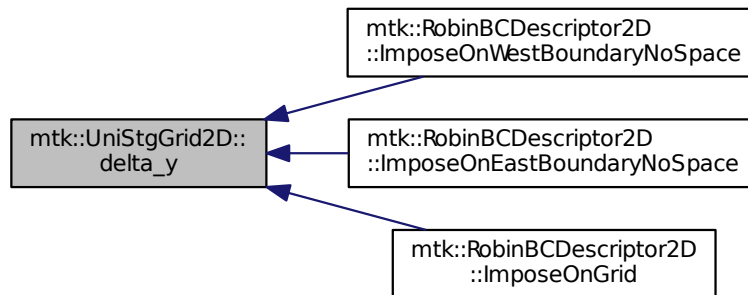
**17.24.3.7 mtk::Real mtk::UniStgGrid2D::delta\_y ( ) const**

## Returns

Computed  $\Delta y$ .

Definition at line 250 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



#### 17.24.3.8 `const mtk::Real * mtk::UniStgGrid2D::discrete_domain_x ( ) const`

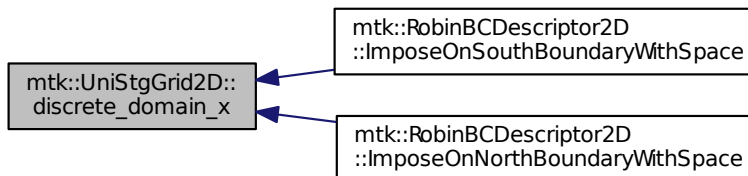
## Returns

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 230 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



#### 17.24.3.9 `const mtk::Real * mtk::UniStgGrid2D::discrete_domain_y ( ) const`

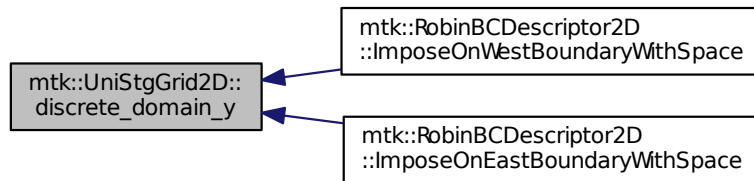
**Returns**

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 260 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



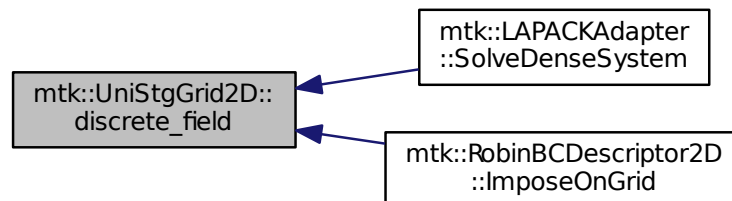
#### 17.24.3.10 `mtk::Real * mtk::UniStgGrid2D::discrete_field ( )`

**Returns**

Pointer to the field data.

Definition at line 265 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



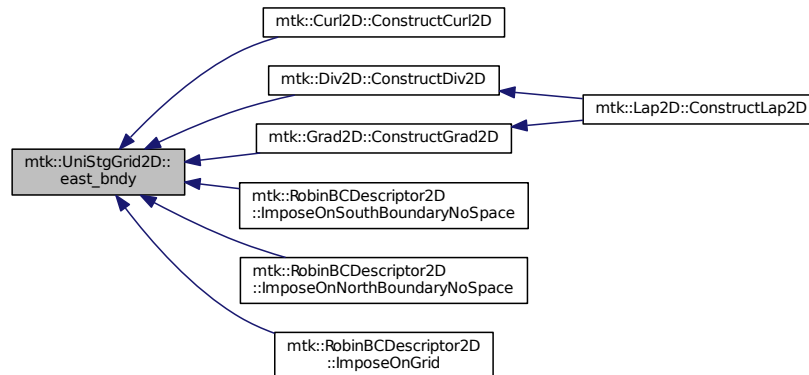
#### 17.24.3.11 `mtk::Real mtk::UniStgGrid2D::east_bndy ( ) const`

## Returns

East boundary spatial coordinate.

Definition at line 215 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



### 17.24.3.12 mtk::FieldNature mtk::UniStgGrid2D::nature ( ) const

## Returns

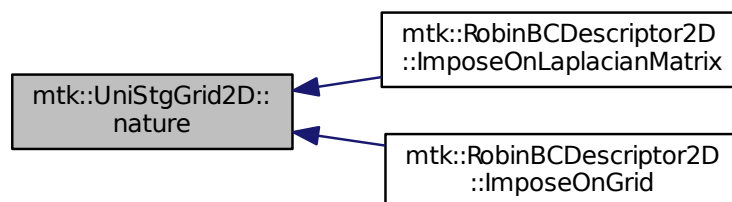
Value of an enumeration.

## See also

[mtk::FieldNature](#)

Definition at line 205 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



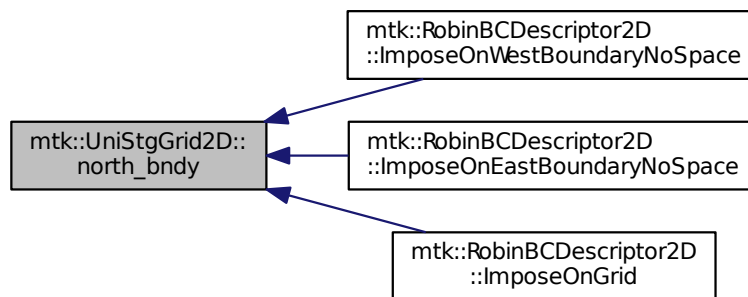
### 17.24.3.13 mtk::Real mtk::UniStgGrid2D::north\_bndy ( ) const

#### Returns

North boundary spatial coordinate.

Definition at line 240 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



### 17.24.3.14 int mtk::UniStgGrid2D::num\_cells\_x ( ) const

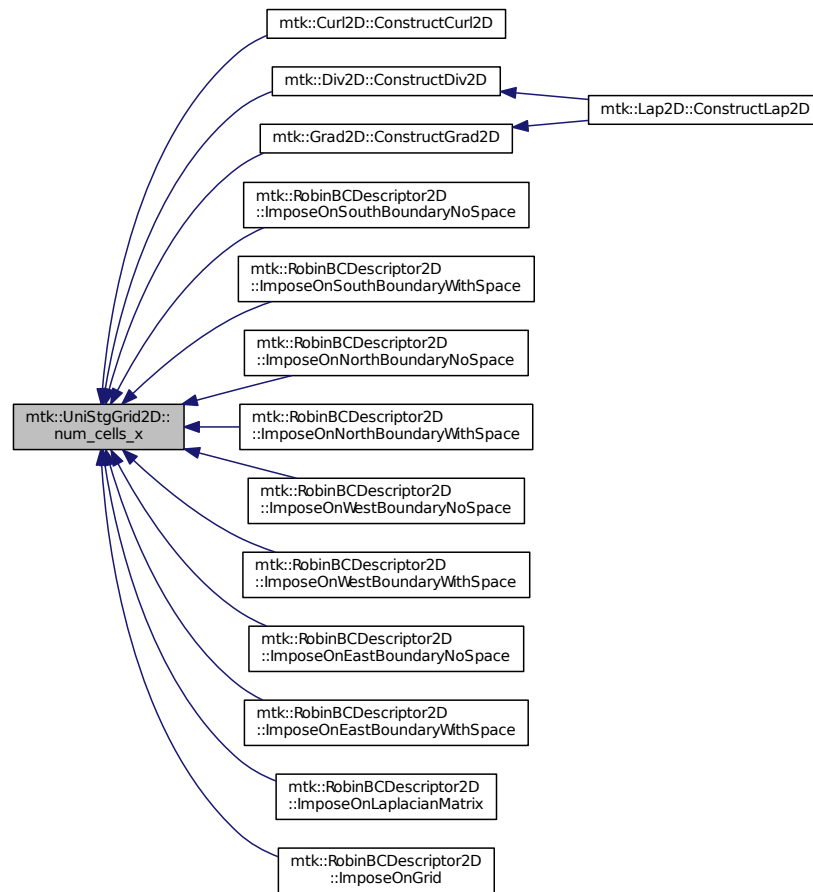
#### Returns

Number of cells of the grid.

Definition at line 220 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).



Here is the caller graph for this function:



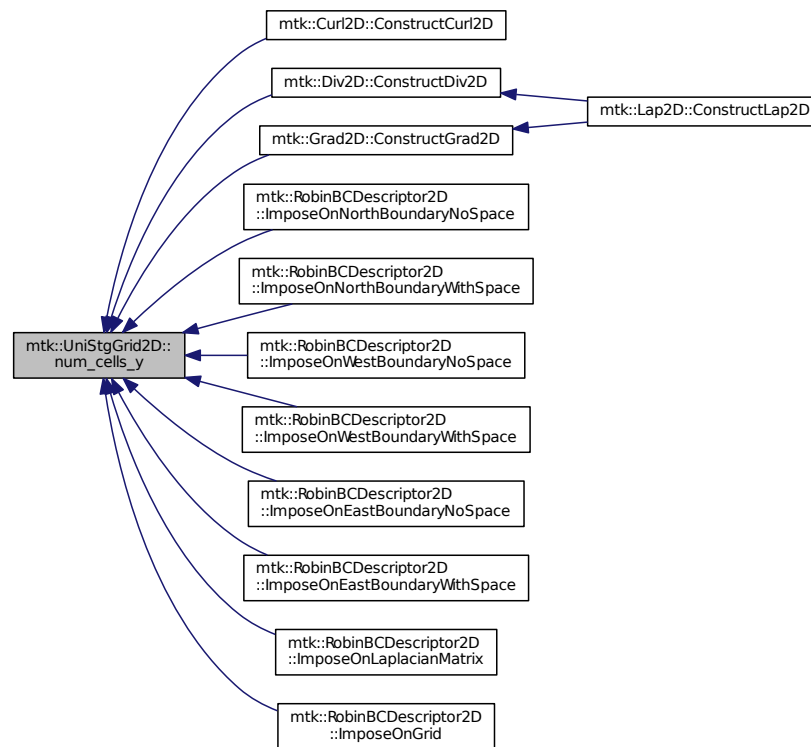
17.24.3.15 `int mtk::UniStgGrid2D::num_cells_y ( ) const`

## Returns

Number of cells of the grid.

Definition at line 245 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



17.24.3.16 `int mtk::UniStgGrid2D::Size ( ) const`

## Returns

Total number of samples in the grid.

Definition at line 270 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

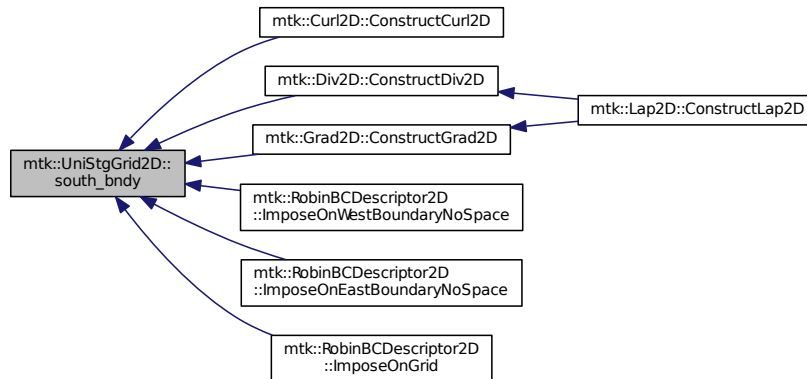
17.24.3.17 `mtk::Real mtk::UniStgGrid2D::south_bndy ( ) const`

**Returns**

South boundary spatial coordinate.

Definition at line 235 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

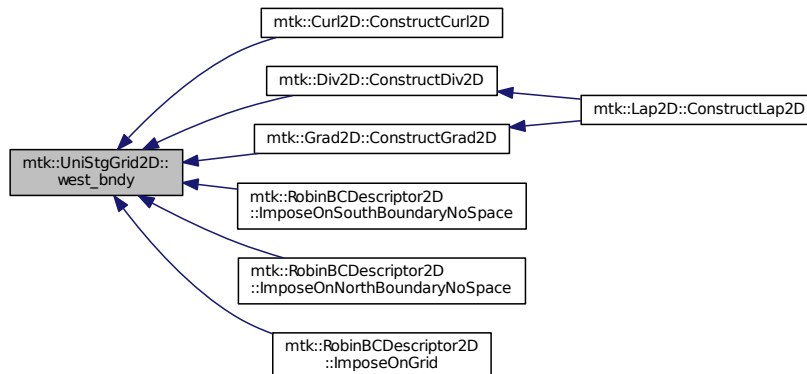
Here is the caller graph for this function:

**17.24.3.18 mtk::Real mtk::UniStgGrid2D::west\_bndy ( ) const****Returns**

West boundary spatial coordinate.

Definition at line 210 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



17.24.3.19 `bool mtk::UniStgGrid2D::WriteToFile ( std::string filename, std::string space_name_x, std::string space_name_y,  
std::string field_name ) const`

## Parameters

in	<i>filename</i>	Name of the output file.
in	<i>space_name_x</i>	Name for the first column of the (spatial) data.
in	<i>space_name_y</i>	Name for the second column of the (spatial) data.
in	<i>field_name</i>	Name for the second column of the (physical field) data.

## Returns

Success of the file writing process.

## See also

<http://www.gnuplot.info/>

Write the values of the p component, with a null q component.

Write the values of the q component, with a null p component.

Definition at line 435 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

## 17.24.4 Friends And Related Function Documentation

17.24.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::UniStgGrid2D & in )` [friend]

1. Print spatial coordinates.

2. Print scalar field.

Definition at line 67 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

## 17.24.5 Member Data Documentation

17.24.5.1 `Real mtk::UniStgGrid2D::delta_x_` [private]

Definition at line 302 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.2 `Real mtk::UniStgGrid2D::delta_y_` [private]

Definition at line 307 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.3 `std::vector<Real> mtk::UniStgGrid2D::discrete_domain_x_` [private]

Definition at line 293 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.4 `std::vector<Real> mtk::UniStgGrid2D::discrete_domain_y_` [private]

Definition at line 294 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.5 `std::vector<Real> mtk::UniStgGrid2D::discrete_field_` [private]

Definition at line 295 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.6 `Real mtk::UniStgGrid2D::east_bndy_` [private]

Definition at line 300 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.7 `FieldNature mtk::UniStgGrid2D::nature_` [private]

Definition at line 297 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.8 `Real mtk::UniStgGrid2D::north_bndy_` [private]

Definition at line 305 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.9 `int mtk::UniStgGrid2D::num_cells_x_` [private]

Definition at line 301 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.10 `int mtk::UniStgGrid2D::num_cells_y_` [private]

Definition at line 306 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.11 `Real mtk::UniStgGrid2D::south_bndy_` [private]

Definition at line 304 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

17.24.5.12 `Real mtk::UniStgGrid2D::west_bndy_` [private]

Definition at line 299 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

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

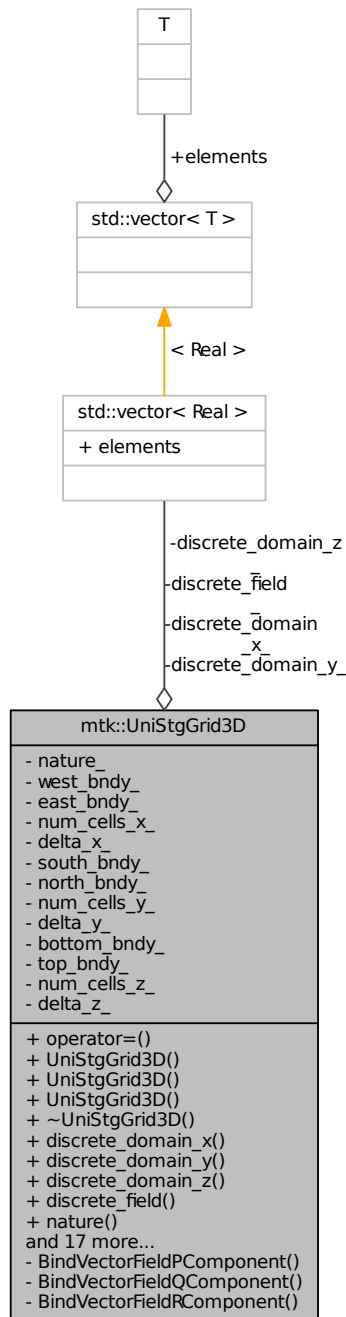
- [include/mtk\\_uni\\_stg\\_grid\\_2d.h](#)
- [src/mtk\\_uni\\_stg\\_grid\\_2d.cc](#)

## 17.25 mtk::UniStgGrid3D Class Reference

Uniform 3D Staggered Grid.

```
#include <mtk_uni_stg_grid_3d.h>
```

Collaboration diagram for mtk::UniStgGrid3D:



## Public Member Functions

- [UniStgGrid3D operator=](#) (const [UniStgGrid3D](#) &in)

*Overloaded assignment operator.*

- [UniStgGrid3D](#) ()

*Default constructor.*

- [UniStgGrid3D](#) (const [UniStgGrid3D](#) &grid)

*Copy constructor.*

- [UniStgGrid3D](#) (const [Real](#) &west\_bndy\_x, const [Real](#) &east\_bndy\_x, const int &num\_cells\_x, const [Real](#) &south\_bndy\_y, const [Real](#) &north\_bndy\_y, const int &num\_cells\_y, const [Real](#) &bottom\_bndy\_z, const [Real](#) &top\_bndy\_z, const int &num\_cells\_z, const [mtk::FieldNature](#) &nature=[mtk::SCALAR](#))

*Construct a grid based on spatial discretization parameters.*

- [~UniStgGrid3D](#) ()

*Destructor.*

- const [Real](#) \* [discrete\\_domain\\_x](#) () const

*Provides access to the grid spatial data.*

- const [Real](#) \* [discrete\\_domain\\_y](#) () const

*Provides access to the grid spatial data.*

- const [Real](#) \* [discrete\\_domain\\_z](#) () const

*Provides access to the grid spatial data.*

- [Real](#) \* [discrete\\_field](#) ()

*Provides access to the grid field data.*

- [FieldNature](#) nature () const

*Physical nature of the data bound to the grid.*

- [Real](#) west\_bndy () const

*Provides access to west boundary spatial coordinate.*

- [Real](#) east\_bndy () const

*Provides access to east boundary spatial coordinate.*

- int num\_cells\_x () const

*Provides access to the number of cells of the grid.*

- [Real](#) delta\_x () const

*Provides access to the computed  $\Delta x$ .*

- [Real](#) south\_bndy () const

*Provides access to south boundary spatial coordinate.*

- [Real](#) north\_bndy () const

*Provides access to north boundary spatial coordinate.*

- int num\_cells\_y () const

*Provides access to the number of cells of the grid.*

- [Real](#) delta\_y () const

*Provides access to the computed  $\Delta y$ .*

- [Real](#) bottom\_bndy () const

*Provides access to bottom boundary spatial coordinate.*

- [Real](#) top\_bndy () const

*Provides access to top boundary spatial coordinate.*

- int num\_cells\_z () const

*Provides access to the number of cells of the grid.*

- [Real](#) delta\_z () const

*Provides access to the computed  $\Delta z$ .*

- bool Bound () const

*Have any field been bound to the grid?*



- int [Size](#) () const  
*Total number of samples in the grid.*
- void [BindScalarField](#) ([Real](#)(\*ScalarField)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz))  
*Binds a given scalar field to the grid.*
- void [BindVectorField](#) ([Real](#)(\*VectorFieldPComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz), [Real](#)(\*VectorFieldQComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz), [Real](#)(\*VectorFieldRComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz))  
*Binds a given vector field to the grid.*
- bool [WriteToFile](#) (std::string filename, std::string space\_name\_x, std::string space\_name\_y, std::string space\_name\_z, std::string field\_name) const  
*Writes grid to a file compatible with Gnuplot 4.6.*

### Private Member Functions

- void [BindVectorFieldPComponent](#) ([Real](#)(\*VectorFieldPComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz))  
*Binds a given component of a vector field to the grid.*
- void [BindVectorFieldQComponent](#) ([Real](#)(\*VectorFieldQComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz))  
*Binds a given component of a vector field to the grid.*
- void [BindVectorFieldRComponent](#) ([Real](#)(\*VectorFieldRComponent)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &zz))  
*Binds a given component of a vector field to the grid.*

### Private Attributes

- std::vector< [Real](#) > [discrete\\_domain\\_x\\_](#)  
*Array of spatial data.*
- std::vector< [Real](#) > [discrete\\_domain\\_y\\_](#)  
*Array of spatial data.*
- std::vector< [Real](#) > [discrete\\_domain\\_z\\_](#)  
*Array of spatial data.*
- std::vector< [Real](#) > [discrete\\_field\\_](#)  
*Array of field's data.*
- [FieldNature](#) [nature\\_](#)  
*Nature of the discrete field.*
- [Real](#) [west\\_bndy\\_](#)  
*West boundary spatial coordinate.*
- [Real](#) [east\\_bndy\\_](#)  
*East boundary spatial coordinate.*
- int [num\\_cells\\_x\\_](#)  
*Number of cells discretizing the domain.*
- [Real](#) [delta\\_x\\_](#)  
*Computed  $\Delta x$ .*
- [Real](#) [south\\_bndy\\_](#)  
*West boundary spatial coordinate.*
- [Real](#) [north\\_bndy\\_](#)

- East boundary spatial coordinate.*
- int [num\\_cells\\_y\\_](#)  
*Number of cells discretizing the domain.*
- [Real delta\\_y\\_](#)  
*Computed  $\Delta y$ .*
- [Real bottom\\_bndy\\_](#)  
*Bottom boundary spatial coordinate.*
- [Real top\\_bndy\\_](#)  
*Top boundary spatial coordinate.*
- int [num\\_cells\\_z\\_](#)  
*Number of cells discretizing the domain.*
- [Real delta\\_z\\_](#)  
*Computed  $\Delta z$ .*

## Friends

- std::ostream & [operator<<](#) (std::ostream &stream, [UniStgGrid3D](#) &in)  
*Prints the grid as a tuple of arrays.*

### 17.25.1 Detailed Description

Uniform 3D Staggered Grid.

Definition at line 79 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

### 17.25.2 Constructor & Destructor Documentation

#### 17.25.2.1 `mtk::UniStgGrid3D::UniStgGrid3D ( )`

Definition at line 123 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

#### 17.25.2.2 `mtk::UniStgGrid3D::UniStgGrid3D ( const UniStgGrid3D & grid )`

##### Parameters

<code>in</code>	<code>grid</code>	Given grid.
-----------------	-------------------	-------------

Definition at line 142 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

#### 17.25.2.3 `mtk::UniStgGrid3D::UniStgGrid3D ( const Real & west_bndy_x, const Real & east_bndy_x, const int & num_cells_x, const Real & south_bndy_y, const Real & north_bndy_y, const int & num_cells_y, const Real & bottom_bndy_z, const Real & top_bndy_z, const int & num_cells_z, const mtk::FieldNature & nature = mtk::SCALAR )`

##### Parameters

in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</i>	Number of cells of the required grid.
in	<i>bottom_bndy_z</i>	Coordinate for the bottom boundary.
in	<i>top_bndy_z</i>	Coordinate for the top boundary.
in	<i>num_cells_z</i>	Number of cells of the required grid.
in	<i>nature</i>	Nature of the discrete field to hold.

See also

[mtk::FieldNature](#)

Definition at line 174 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the call graph for this function:



#### 17.25.2.4 mtk::UniStgGrid3D::~~UniStgGrid3D ( )

Definition at line 221 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

### 17.25.3 Member Function Documentation

#### 17.25.3.1 void mtk::UniStgGrid3D::BindScalarField ( Real(\*) (const Real &xx, const Real &yy, const Real &zz) *ScalarField* )

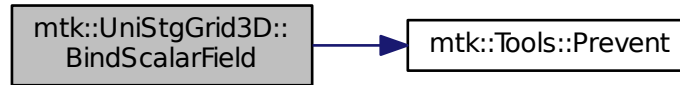
Parameters

in	<i>ScalarField</i>	Pointer to the function implementing the scalar field.
----	--------------------	--------------------------------------------------------

1. Create collection of spatial coordinates for  $x$ .
2. Create collection of spatial coordinates for  $y$ .
3. Create collection of spatial coordinates for  $z$ .
4. Create collection of field samples.

Definition at line 318 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the call graph for this function:



17.25.3.2 void mtk::UniStgGrid3D::BindVectorField ( Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldPComponent, Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldQComponent, Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldRComponent )

We assume the field to be of the form:

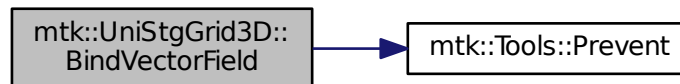
$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

#### Parameters

in	<i>VectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
in	<i>VectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
in	<i>VectorFieldRComponent</i>	Pointer to the function implementing the \$ r \$ component of the vector field.

Definition at line 414 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the call graph for this function:



17.25.3.3 void mtk::UniStgGrid3D::BindVectorFieldPComponent ( Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldPComponent ) [private]

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

## Parameters

in	<i>BindVectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
----	----------------------------------	---------------------------------------------------------------------------------

Definition at line 393 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.4 void mtk::UniStgGrid3D::BindVectorFieldQComponent ( Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldQComponent ) [private]

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

## Parameters

in	<i>BindVectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
----	----------------------------------	---------------------------------------------------------------------------------

Definition at line 400 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.5 void mtk::UniStgGrid3D::BindVectorFieldRComponent ( Real(\*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldRComponent ) [private]

We assume the field to be of the form:

$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

## Parameters

in	<i>BindVectorFieldRComponent</i>	Pointer to the function implementing the \$ r \$ component of the vector field.
----	----------------------------------	---------------------------------------------------------------------------------

Definition at line 407 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

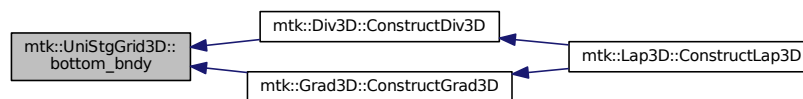
17.25.3.6 mtk::Real mtk::UniStgGrid3D::bottom\_bndy ( ) const

## Returns

Bottom boundary spatial coordinate.

Definition at line 278 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



17.25.3.7 `bool mtk::UniStgGrid3D::Bound ( ) const`

Returns

True is a field has been bound.

Definition at line 308 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.8 `mtk::Real mtk::UniStgGrid3D::delta_x ( ) const`

Returns

Computed  $\Delta x$ .

Definition at line 243 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.9 `mtk::Real mtk::UniStgGrid3D::delta_y ( ) const`

Returns

Computed  $\Delta y$ .

Definition at line 268 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.10 `mtk::Real mtk::UniStgGrid3D::delta_z ( ) const`

Returns

Computed  $\Delta z$ .

Definition at line 293 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.11 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_x ( ) const`

Returns

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 248 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.12 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_y ( ) const`

Returns

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 273 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.13 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_z ( ) const`

#### Returns

Pointer to the spatial data.

**Todo** Review const-correctness of the pointer we return.

Definition at line 298 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.14 `mtk::Real * mtk::UniStgGrid3D::discrete_field ( )`

#### Returns

Pointer to the field data.

Definition at line 303 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

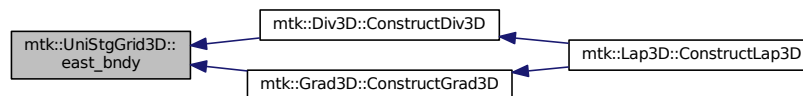
17.25.3.15 `mtk::Real mtk::UniStgGrid3D::east_bndy ( ) const`

#### Returns

East boundary spatial coordinate.

Definition at line 233 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



17.25.3.16 `mtk::FieldNature mtk::UniStgGrid3D::nature ( ) const`

#### Returns

Value of an enumeration.

#### See also

[mtk::FieldNature](#)

Definition at line 223 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.17 `mtk::Real mtk::UniStgGrid3D::north_bndy ( ) const`

#### Returns

North boundary spatial coordinate.

Definition at line 258 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

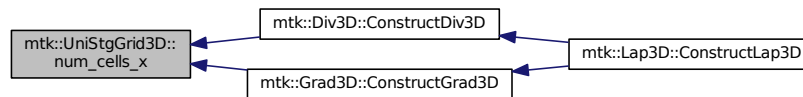
17.25.3.18 `int mtk::UniStgGrid3D::num_cells_x ( ) const`

#### Returns

Number of cells of the grid.

Definition at line 238 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



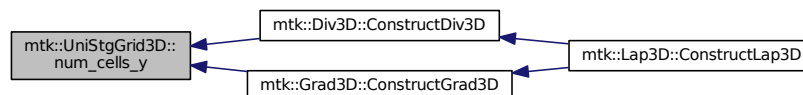
17.25.3.19 `int mtk::UniStgGrid3D::num_cells_y ( ) const`

#### Returns

Number of cells of the grid.

Definition at line 263 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



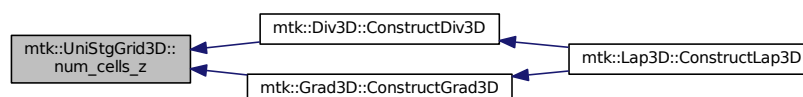
17.25.3.20 `int mtk::UniStgGrid3D::num_cells_z ( ) const`

#### Returns

Number of cells of the grid.

Definition at line 288 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:





17.25.3.21 `mtk::UniStgGrid3D mtk::UniStgGrid3D::operator= ( const UniStgGrid3D & in )`

#### Parameters

<i>in</i>	<i>in</i>	Given grid.
-----------	-----------	-------------

#### Returns

Copy of the given grid.

Definition at line 116 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

17.25.3.22 `int mtk::UniStgGrid3D::Size ( ) const`

#### Returns

Total number of samples in the grid.

Definition at line 313 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

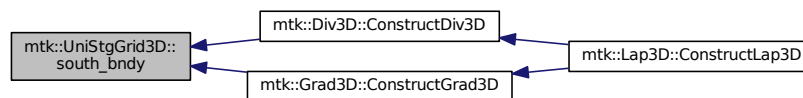
17.25.3.23 `mtk::Real mtk::UniStgGrid3D::south_bndy ( ) const`

#### Returns

South boundary spatial coordinate.

Definition at line 253 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



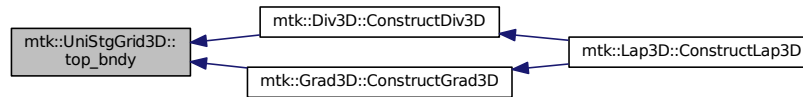
17.25.3.24 `mtk::Real mtk::UniStgGrid3D::top_bndy ( ) const`

**Returns**

Top boundary spatial coordinate.

Definition at line 283 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



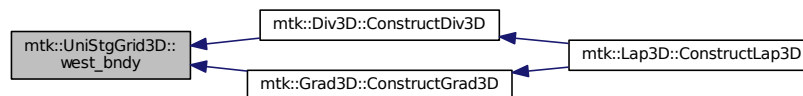
### 17.25.3.25 `mtk::Real mtk::UniStgGrid3D::west_bndy ( ) const`

**Returns**

West boundary spatial coordinate.

Definition at line 228 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

Here is the caller graph for this function:



### 17.25.3.26 `bool mtk::UniStgGrid3D::WriteToFile ( std::string filename, std::string space_name_x, std::string space_name_y, std::string space_name_z, std::string field_name ) const`

**Parameters**

in	<i>filename</i>	Name of the output file.
in	<i>space_name_x</i>	Name for the first column of the (spatial) data.
in	<i>space_name_y</i>	Name for the second column of the (spatial) data.
in	<i>space_name_z</i>	Name for the third column of the (spatial) data.
in	<i>field_name</i>	Name for the second column of the (physical field) data.

**Returns**

Success of the file writing process.

See also

<http://www.gnuplot.info/>

Definition at line 433 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

## 17.25.4 Friends And Related Function Documentation

17.25.4.1 `std::ostream& operator<< ( std::ostream & stream, mtk::UniStgGrid3D & in )` [friend]

1. Print spatial coordinates.

2. Print scalar field.

Definition at line 67 of file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

## 17.25.5 Member Data Documentation

17.25.5.1 `Real mtk::UniStgGrid3D::bottom_bndy_` [private]

Definition at line 396 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.2 `Real mtk::UniStgGrid3D::delta_x_` [private]

Definition at line 389 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.3 `Real mtk::UniStgGrid3D::delta_y_` [private]

Definition at line 394 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.4 `Real mtk::UniStgGrid3D::delta_z_` [private]

Definition at line 399 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.5 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_x_` [private]

Definition at line 379 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.6 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_y_` [private]

Definition at line 380 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.7 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_z_` [private]

Definition at line 381 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.8 `std::vector<Real> mtk::UniStgGrid3D::discrete_field_` [private]

Definition at line 382 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.9 `Real mtk::UniStgGrid3D::east_bndy_` [private]

Definition at line 387 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.10 `FieldNature mtk::UniStgGrid3D::nature_` [private]

Definition at line 384 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.11 `Real mtk::UniStgGrid3D::north_bndy_` [private]

Definition at line 392 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.12 `int mtk::UniStgGrid3D::num_cells_x_` [private]

Definition at line 388 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.13 `int mtk::UniStgGrid3D::num_cells_y_` [private]

Definition at line 393 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.14 `int mtk::UniStgGrid3D::num_cells_z_` [private]

Definition at line 398 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.15 `Real mtk::UniStgGrid3D::south_bndy_` [private]

Definition at line 391 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.16 `Real mtk::UniStgGrid3D::top_bndy_` [private]

Definition at line 397 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

17.25.5.17 `Real mtk::UniStgGrid3D::west_bndy_` [private]

Definition at line 386 of file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

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

- [include/mtk\\_uni\\_stg\\_grid\\_3d.h](#)
- [src/mtk\\_uni\\_stg\\_grid\\_3d.cc](#)

## Chapter 18

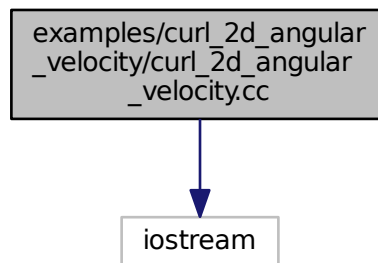
# File Documentation

### 18.1 examples/curl\_2d\_angular\_velocity/curl\_2d\_angular\_velocity.cc File Reference

Compute the curl of a 2D angular velocity field.

```
#include <iostream>
```

Include dependency graph for curl\_2d\_angular\_velocity.cc:



#### Functions

- int `main` ()

#### 18.1.1 Detailed Description

We compute the curl of:

$$\mathbf{v}(x,y) = -y\hat{\mathbf{i}} + x\hat{\mathbf{j}}.$$

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [curl\\_2d\\_angular\\_velocity.cc](#).

**18.1.2 Function Documentation****18.1.2.1 int main ( )**

Definition at line 106 of file [curl\\_2d\\_angular\\_velocity.cc](#).

**18.2 curl\_2d\_angular\_velocity.cc**

```
00001
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #if __cplusplus == 201103L
00060
00061 #include <iostream>
00062 #include <fstream>
00063 #include <cmath>
00064
00065 #include <vector>
00066
00067 #include "mtk.h"
00068
```

```

00069 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const
 mtk::Real &yy) {
00070
00071 return -yy;
00072 }
00073
00074 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
 mtk::Real &yy) {
00075
00076 return xx;
00077 }
00078
00079 int main () {
00080
00081 std::cout << "Example: Curl of a angular velocity field." << std::endl;
00082
00083 mtk::Real aa = 0.0;
00084 mtk::Real bb = 4.0;
00085 mtk::Real cc = 0.0;
00086 mtk::Real dd = 4.0;
00087
00088 int nn = 10;
00089 int mm = 10;
00090
00091 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00092
00093 gg.BindVectorField(VectorFieldPComponent, VectorFieldQComponent);
00094
00095 if(!gg.WriteToFile("curl_2d_angular_velocity_gg.dat", "x", "y", "v(x,y)")) {
00096 std::cerr << "Angular field could not be written to disk." << std::endl;
00097 return EXIT_FAILURE;
00098 }
00099 }
00100
00101 #else
00102 #include <iostream>
00103 using std::cout;
00104 using std::endl;
00105 int main () {
00106 cout << "This code HAS to be compiled with support for C++11." << endl;
00107 cout << "Exiting..." << endl;
00108 return EXIT_SUCCESS;
00109 }
00110 #endif

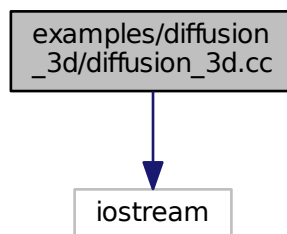
```

## 18.3 examples/diffusion\_3d/diffusion\_3d.cc File Reference

Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs.

```
#include <iostream>
```

Include dependency graph for diffusion\_3d.cc:



## Functions

- `int main ()`

### 18.3.1 Detailed Description

We solve:

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{x}),$$

for  $\mathbf{x} \in \Omega = [0, 1]^3$ .

We consider autonomous homogeneous Dirichlet boundary conditions.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [diffusion\\_3d.cc](#).

### 18.3.2 Function Documentation

#### 18.3.2.1 `int main ( )`

Definition at line 123 of file [diffusion\\_3d.cc](#).

## 18.4 `diffusion_3d.cc`

```

00001
00016 /*
00017 Copyright (C) 2015, Computational Science Research Center, San Diego State
00018 University. All rights reserved.
00019
00020 Redistribution and use in source and binary forms, with or without modification,
00021 are permitted provided that the following conditions are met:
00022
00023 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00024 and a copy of the modified files should be reported once modifications are
00025 completed, unless these modifications are made through the project's GitHub
00026 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00027 should be developed and included in any deliverable.
00028
00029 2. Redistributions of source code must be done through direct
00030 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00031
00032 3. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 4. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders, and due credit should
00038 be given to the copyright holders.
00039
00040 5. Neither the name of the copyright holder nor the names of its contributors
00041 may be used to endorse or promote products derived from this software without
00042 specific prior written permission.
00043
00044 The copyright holders provide no reassurances that the source code provided does
00045 not infringe any patent, copyright, or any other intellectual property rights of
00046 third parties. The copyright holders disclaim any liability to any recipient for
00047 claims brought against recipient by any third party for infringement of that
00048 parties intellectual property rights.
00049
00050 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

```



```

00051 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00052 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00053 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00054 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00055 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00056 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00057 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00058 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00059 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00060 */
00061
00062 #if __cplusplus == 201103L
00063
00064 #include <iostream>
00065 #include <fstream>
00066 #include <cmath>
00067
00068 #include <vector>
00069
00070 #include "mtk.h"
00071
00072 mtk::Real InitialCondition(const mtk::Real &xx,
00073 const mtk::Real &yy,
00074 const mtk::Real &zz) {
00075
00076 mtk::Real rr{0.3};
00077
00078 mtk::Real aux{xx*xx + yy*yy + zz*zz};
00079
00080 return (aux < rr? rr - aux: mtk::kZero);
00081 }
00082
00083 int main () {
00084
00085 std::cout << "Example: Diffusion Equation in 3D "
00086 "with Dirichlet BCs." << std::endl;
00087
00088 mtk::Real west_bndy_x{0.0};
00089 mtk::Real east_bndy_x{1.0};
00090 mtk::Real south_bndy_y{0.0};
00091 mtk::Real north_bndy_y{1.0};
00092 mtk::Real bottom_bndy_z{0.0};
00093 mtk::Real top_bndy_z{1.0};
00094
00095 int num_cells_x{50};
00096 int num_cells_y{50};
00097 int num_cells_z{50};
00098
00099 mtk::UniStgGrid3D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00100 south_bndy_y, north_bndy_y, num_cells_y,
00101 bottom_bndy_z, top_bndy_z, num_cells_z);
00102
00103 comp_sol.BindScalarField(InitialCondition);
00104
00105 if(!comp_sol.WriteToFile("diffusion_3d_comp_sol.dat",
00106 "x",
00107 "y",
00108 "z",
00109 "Initial u(x,y,z)")) {
00110 std::cerr << "Error writing to file." << std::endl;
00111 return EXIT_FAILURE;
00112 }
00113
00114 #else
00115 #include <iostream>
00116 using std::cout;
00117 using std::endl;
00118 int main () {
00119 cout << "This code HAS to be compiled with support for C++11." << endl;
00120 cout << "Exiting..." << endl;
00121 return EXIT_SUCCESS;
00122 }
00123 #endif

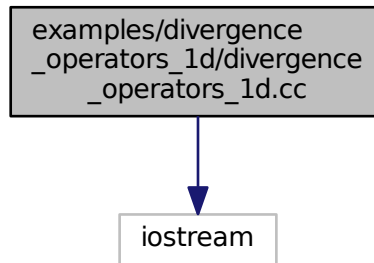
```

## 18.5 examples/divergence\_operators\_1d/divergence\_operators\_1d.cc File Reference

Creates instances of a 1D divergence as computed by the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for divergence\_operators\_1d.cc:



### Functions

- int [main](#) ()

#### 18.5.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [divergence\\_operators\\_1d.cc](#).

#### 18.5.2 Function Documentation

##### 18.5.2.1 int main ( )

Definition at line [102](#) of file [divergence\\_operators\\_1d.cc](#).

## 18.6 divergence\_operators\_1d.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are

```

```

00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066 std::cout << "Example: Instances of a 1D divergence as computed by the CBS "
00067 "algorithm." << std::endl;
00068
00069 std::ofstream output_tex_file;
00070
00071 int max_order{6};
00072
00073 for (int order = 2; order <= max_order; order += 2) {
00074
00075 std::string output_tex_file_name{"div_1d_" + std::to_string(order) +
00076 ".tex"};
00077
00078 output_tex_file.open(output_tex_file_name);
00079
00080 mtk::Div1D div;
00081
00082 bool assertion = div.ConstructDiv1D(order);
00083
00084 if (!assertion) {
00085 std::cerr << "Mimetic div (order" + std::to_string(order) +
00086 ") could not be built." << std::endl;
00087 return EXIT_FAILURE;
00088 }
00089
00090 output_tex_file << "\\begin{verbatim}" << std::endl;
00091 output_tex_file << div << std::endl;
00092 output_tex_file << "\\end{verbatim}" << std::endl;
00093 output_tex_file.close();
00094 }
00095 }
00096
00097 #else
00098

```

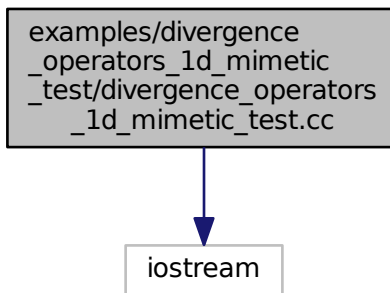
```
00099 #include <iostream>
00100 using std::cout;
00101 using std::endl;
00102 int main () {
00103 cout << "This code HAS to be compiled with support for C++11." << endl;
00104 cout << "Exiting..." << endl;
00105 return EXIT_SUCCESS;
00106 }
00107 #endif
```

## 18.7 examples/divergence\_operators\_1d\_mimetic\_test/divergence\_operators\_1d\_mimetic\_test.cc File Reference

Test mimetic qualities of instances of a 1D divergence from the CBSA.

```
#include <iostream>
```

Include dependency graph for divergence\_operators\_1d\_mimetic\_test.cc:



### Functions

- int `main` ()

#### 18.7.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [divergence\\_operators\\_1d\\_mimetic\\_test.cc](#).

#### 18.7.2 Function Documentation

##### 18.7.2.1 int main ( )

Definition at line 101 of file [divergence\\_operators\\_1d\\_mimetic\\_test.cc](#).

## 18.8 divergence\_operators\_1d\_mimetic\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066 std::cout << "Example: Instances of a 1D divergence as computed by the CBS "
00067 "algorithm." << std::endl;
00068
00069
00070
00071 std::ofstream output_tex_file;
00072
00073 output_tex_file.open("div_1d_mim_test.tex");
00074
00075 int max_order{14};
00076
00077 for (int order = 2; order <= max_order; order += 2) {
00078
00079 mtk::Div1D div;
00080
00081 bool assertion = div.ConstructDiv1D(order);
00082 if (!assertion) {
00083 std::cerr << "Mimetic div (order" + std::to_string(order) +
00084 ") could not be built." << std::endl;
00085 return EXIT_FAILURE;

```

```

00086 }
00087
00088 int num_cells_x{3*order - 1};
00089
00090 mtk::DenseMatrix divm(div.ReturnAsDimensionlessDenseMatrix
(num_cells_x));
00091
00092 std::cout << order << ' ' << divm.MaxFromSumsOfRowElements() << std::endl;
00093 getchar();
00094 }
00095 }
00096
00097 #else
00098 #include <iostream>
00099 using std::cout;
00100 using std::endl;
00101 int main () {
00102 cout << "This code HAS to be compiled with support for C++11." << endl;
00103 cout << "Exiting..." << endl;
00104 return EXIT_SUCCESS;
00105 }
00106 #endif

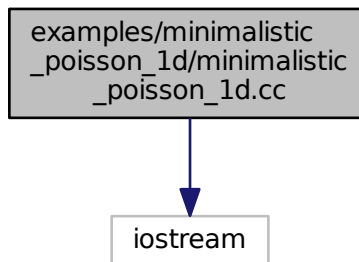
```

## 18.9 examples/minimalistic\_poisson\_1d/minimalistic\_poisson\_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for minimalistic\_poisson\_1d.cc:



### Functions

- int `main` ()

### 18.9.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for  $x \in \Omega = [a, b] = [0, 1]$ .

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where  $\lambda = -1$  is a real-valued parameter.

We consider Robin's boundary conditions of the form:

$$\alpha p(a) - \beta p'(a) = \omega,$$

$$\alpha p(b) + \beta p'(b) = \varepsilon,$$

where  $\alpha = -\exp(\lambda)$ ,  $\beta = (\exp(\lambda) - 1.0)/\lambda$ ,  $\omega = -1$ , and  $\varepsilon = 0$ .

The analytical solution for this problem is given by:

$$p(x) = \frac{\exp(\lambda x) - 1}{\exp(\lambda) - 1}.$$

The mimetic counterpart of this equation is:

$$-\check{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering  $k = 2$ .

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [minimalistic\\_poisson\\_1d.cc](#).

## 18.9.2 Function Documentation

### 18.9.2.1 int main ( )

Definition at line 164 of file [minimalistic\\_poisson\\_1d.cc](#).

## 18.10 minimalistic\_poisson\_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066

```

```

00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #if __cplusplus == 201103L
00090
00091 #include <iostream>
00092 #include <fstream>
00093 #include <cmath>
00094 #include <vector>
00095
00096 #include "mtk.h"
00097
00098 mtk::Real Alpha(const mtk::Real &tt) {
00099 mtk::Real lambda = -1.0;
00100 return -exp(lambda);
00101 }
00102
00103 mtk::Real Beta(const mtk::Real &tt) {
00104 mtk::Real lambda = -1.0;
00105 return (exp(lambda) - 1.0)/lambda;
00106 };
00107
00108 mtk::Real Omega(const mtk::Real &tt) { return -1.0; };
00109
00110 mtk::Real Epsilon(const mtk::Real &tt) { return 0.0; };
00111
00112 mtk::Real Source(const mtk::Real &xx) {
00113 mtk::Real lambda = -1.0;
00114 return lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00115 }
00116
00117 mtk::Real KnownSolution(const mtk::Real &xx) {
00118 mtk::Real lambda = -1.0;
00119 return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00120 }
00121
00122 int main () {
00123
00124 mtk::Real west_bndy_x{};
00125 mtk::Real east_bndy_x{1.0};
00126 int num_cells_x{5};
00127 mtk::LaplD lap;
00128 if (!lap.ConstructLaplD()) {
00129 return EXIT_FAILURE;
00130 }
00131 mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00132 mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00133 mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00134 mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00135 source.BindScalarField(Source);
00136 mtk::RobinBCDescriptor1D bcs;
00137 bcs.PushBackWestCoeff(Alpha);
00138 bcs.PushBackWestCoeff(Beta);
00139 bcs.PushBackEastCoeff(Alpha);
00140 bcs.PushBackEastCoeff(Beta);
00141 bcs.set_west_condition(Omega);
00142 bcs.set_east_condition(Epsilon);
00143 if (!bcs.ImposeOnLaplacianMatrix(lap, lapm)) {
00144 return EXIT_FAILURE;
00145 }
00146 bcs.ImposeOnGrid(source);
00147 int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};

```



```

00148 if (info != 0) {
00149 return EXIT_FAILURE;
00150 }
00151 source.WriteToFile("minimalistic_poisson_1d_comp_sol.dat", "x", "~u(x)");
00152 known_sol.BindScalarField(KnownSolution);
00153 known_sol.WriteToFile("minimalistic_poisson_1d_known_sol.dat", "x", "u(x)");
00154 mtk::Real relative_norm_2_error =
00155 mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00156 known_sol.discrete_field(),
00157 known_sol.num_cells_x());
00158 std::cout << relative_norm_2_error << std::endl;
00159 }
00160 #else
00161 #include <iostream>
00162 using std::cout;
00163 using std::endl;
00164 int main () {
00165 cout << "This code HAS to be compiled with support for C++11." << endl;
00166 cout << "Exiting..." << endl;
00167 return EXIT_SUCCESS;
00168 }
00169 #endif

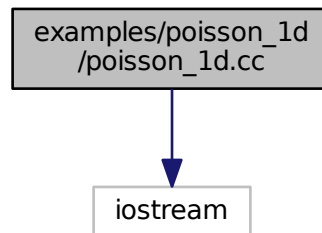
```

## 18.11 examples/poisson\_1d/poisson\_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for poisson\_1d.cc:



### Functions

- int `main` ()

#### 18.11.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for  $x \in \Omega = [a, b] = [0, 1]$ .

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where  $\lambda = -1$  is a real-valued parameter.

We consider Robin's boundary conditions of the form:

$$\alpha p(a) - \beta p'(a) = \omega,$$

$$\alpha p(b) + \beta p'(b) = \varepsilon,$$

where  $\alpha = -\exp(\lambda)$ ,  $\beta = (\exp(\lambda) - 1.0)/\lambda$ ,  $\omega = -1$ , and  $\varepsilon = 0$ .

The analytical solution for this problem is given by:

$$p(x) = \frac{\exp(\lambda x) - 1}{\exp(\lambda) - 1}.$$

The mimetic counterpart of this equation is:

$$-\mathbf{\check{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering  $k = 2$ .

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [poisson\\_1d.cc](#).

### 18.11.2 Function Documentation

#### 18.11.2.1 int main ( )

Definition at line 263 of file [poisson\\_1d.cc](#).

## 18.12 poisson\_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does

```

```

00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #if __cplusplus == 201103L
00090
00091 #include <iostream>
00092 #include <fstream>
00093 #include <cmath>
00094
00095 #include <vector>
00096
00097 #include "mtk.h"
00098
00099 mtk::Real Alpha(const mtk::Real &tt) {
00100 mtk::Real lambda{-1.0};
00101 return -exp(lambda);
00102 }
00103
00104 mtk::Real Beta(const mtk::Real &tt) {
00105 mtk::Real lambda{-1.0};
00106 return (exp(lambda) - 1.0)/lambda;
00107 };
00108
00109 mtk::Real Omega(const mtk::Real &tt) {
00110 return -1.0;
00111 };
00112
00113 mtk::Real Epsilon(const mtk::Real &tt) {
00114 return 0.0;
00115 };
00116
00117 mtk::Real Source(const mtk::Real &xx) {
00118 mtk::Real lambda{-1.0};
00119 return -lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00120 }
00121
00122 mtk::Real KnownSolution(const mtk::Real &xx) {
00123 mtk::Real lambda{-1.0};
00124 return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00125 }
00126
00127 int main () {
00128 std::cout << "Example: Poisson Equation with Robin BCs on a";
00129 std::cout << "1D Uniform Staggered Grid." << std::endl;
00130
00131 mtk::Real west_bndy_x{0.0};
00132 mtk::Real east_bndy_x{1.0};
00133 int num_cells_x{50};
00134
00135 mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00136
00137 mtk::Lap1D lap;
00138
00139 if (!lap.ConstructLap1D()) {
00140 std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00141 return EXIT_FAILURE;
00142 }

```

```

00155 }
00156
00157 std::cout << "lap=" << std::endl;
00158 std::cout << lap << std::endl;
00159
00160 mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00161
00162 std::cout << "lapm =" << std::endl;
00163 std::cout << lapm << std::endl;
00164
00165 lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00166
00167 std::cout << "-lapm =" << std::endl;
00168 std::cout << lapm << std::endl;
00169
00170 mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00171
00172 source.BindScalarField(Source);
00173
00174 std::cout << "source =" << std::endl;
00175 std::cout << source << std::endl;
00176
00177 mtk::RobinBCDescriptor1D robin_bc_desc_ld;
00178
00179 robin_bc_desc_ld.PushBackWestCoeff(Alpha);
00180 robin_bc_desc_ld.PushBackWestCoeff(Beta);
00181
00182 robin_bc_desc_ld.PushBackEastCoeff(Alpha);
00183 robin_bc_desc_ld.PushBackEastCoeff(Beta);
00184
00185 robin_bc_desc_ld.set_west_condition(Omega);
00186 robin_bc_desc_ld.set_east_condition(Epsilon);
00187
00188 if (!robin_bc_desc_ld.ImposeOnLaplacianMatrix(lap, lapm)) {
00189 std::cerr << "BCs could not be bound to the matrix." << std::endl;
00190 return EXIT_FAILURE;
00191 }
00192
00193 std::cout << "Mimetic Laplacian operator with imposed BCs:" << std::endl;
00194 std::cout << lapm << std::endl;
00195
00196 if (!lapm.WriteToFile("poisson_ld_lapm.dat")) {
00197 std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00198 return EXIT_FAILURE;
00199 }
00200
00201 robin_bc_desc_ld.ImposeOnGrid(source);
00202
00203 std::cout << "source =" << std::endl;
00204 std::cout << source << std::endl;
00205
00206 if (!source.WriteToFile("poisson_ld_source.dat", "x", "s(x)")) {
00207 std::cerr << "Source term could not be written to disk." << std::endl;
00208 return EXIT_FAILURE;
00209 }
00210
00211 int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00212
00213 if (!info) {
00214 std::cout << "System solved." << std::endl;
00215 std::cout << std::endl;
00216 } else {
00217 std::cerr << "Something wrong solving system! info = " << info << std::endl;
00218 std::cerr << "Exiting..." << std::endl;
00219 return EXIT_FAILURE;
00220 }
00221
00222 std::cout << "Computed solution:" << std::endl;
00223 std::cout << source << std::endl;
00224
00225 if (!source.WriteToFile("poisson_ld_comp_sol.dat", "x", "~u(x)")) {
00226 std::cerr << "Solution could not be written to file." << std::endl;
00227 return EXIT_FAILURE;
00228 }
00229
00230 mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00231
00232 known_sol.BindScalarField(KnownSolution);
00233
00234 std::cout << "known_sol =" << std::endl;

```

```

00242 std::cout << known_sol << std::endl;
00243
00244 if (!known_sol.WriteToFile("poisson_1d_known_sol.dat", "x", "u(x)")) {
00245 std::cerr << "Known solution could not be written to file." << std::endl;
00246 return EXIT_FAILURE;
00247 }
00248
00249 mtk::Real relative_norm_2_error{};
00250
00251 relative_norm_2_error =
00252 mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00253 known_sol.discrete_field(),
00254 known_sol.num_cells_x());
00255
00256 std::cout << "relative_norm_2_error = ";
00257 std::cout << relative_norm_2_error << std::endl;
00258 }
00259 #else
00260 #include <iostream>
00261 using std::cout;
00262 using std::endl;
00263 int main () {
00264 cout << "This code HAS to be compiled with support for C++11." << endl;
00265 cout << "Exiting..." << endl;
00266 return EXIT_SUCCESS;
00267 }
00268 #endif

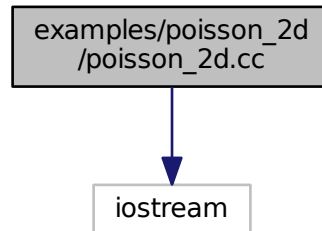
```

## 18.13 examples/poisson\_2d/poisson\_2d.cc File Reference

Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for poisson\_2d.cc:



### Functions

- int `main` ()

#### 18.13.1 Detailed Description

We solve:

$$\nabla^2 u(\mathbf{x}) = s(\mathbf{x}),$$

for  $\mathbf{x} \in \Omega = [0, 1]^2$ .

The source term function is defined as

$$s(x, y) = xye^{-0.5(x^2+y^2)}(x^2+y^2-6).$$

Let  $\partial\Omega = S \cup N \cup W \cup E$ . We consider Dirichlet boundary conditions of the following form:

$$\forall \mathbf{x} \in W : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in E : u(1, y) = -e^{-0.5(1-y^2)}(1-y^2).$$

$$\forall \mathbf{x} \in S : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in N : u(x, 1) = -e^{-0.5(x^2-1)}(x^2-1).$$

The analytical solution for this problem is given by

$$u(x, y) = xye^{-0.5(x^2+y^2)}.$$

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [poisson\\_2d.cc](#).

## 18.13.2 Function Documentation

### 18.13.2.1 int main ( )

Definition at line 241 of file [poisson\\_2d.cc](#).

## 18.14 poisson\_2d.cc

```

00001
00039 /*
00040 Copyright (C) 2015, Computational Science Research Center, San Diego State
00041 University. All rights reserved.
00042
00043 Redistribution and use in source and binary forms, with or without modification,
00044 are permitted provided that the following conditions are met:
00045
00046 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00047 and a copy of the modified files should be reported once modifications are
00048 completed, unless these modifications are made through the project's GitHub
00049 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00050 should be developed and included in any deliverable.
00051
00052 2. Redistributions of source code must be done through direct
00053 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00054
00055 3. Redistributions in binary form must reproduce the above copyright notice,
00056 this list of conditions and the following disclaimer in the documentation and/or
00057 other materials provided with the distribution.
00058
00059 4. Usage of the binary form on proprietary applications shall require explicit
00060 prior written permission from the the copyright holders, and due credit should
00061 be given to the copyright holders.
00062
00063 5. Neither the name of the copyright holder nor the names of its contributors
00064 may be used to endorse or promote products derived from this software without
00065 specific prior written permission.
00066
00067 The copyright holders provide no reassurances that the source code provided does

```

```

00068 not infringe any patent, copyright, or any other intellectual property rights of
00069 third parties. The copyright holders disclaim any liability to any recipient for
00070 claims brought against recipient by any third party for infringement of that
00071 parties intellectual property rights.
00072
00073 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00074 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00075 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00076 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00077 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00078 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00079 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00080 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00081 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00082 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00083 */
00084
00085 #if __cplusplus == 201103L
00086
00087 #include <iostream>
00088 #include <fstream>
00089 #include <cmath>
00090
00091 #include <vector>
00092
00093 #include "mtk.h"
00094
00095 mtk::Real Source(const mtk::Real &xx, const mtk::Real &yy) {
00096
00097 mtk::Real x_squared{xx*xx};
00098 mtk::Real y_squared{yy*yy};
00099 mtk::Real aux{-0.5*(x_squared + y_squared)};
00100
00101 return xx*yy*exp(aux)*(x_squared + y_squared - 6.0);
00102 }
00103
00104 mtk::Real BCCoeff(const mtk::Real &xx, const mtk::Real &yy) {
00105
00106 return mtk::kOne;
00107 }
00108
00109 mtk::Real WestBC(const mtk::Real &xx, const mtk::Real &tt) {
00110
00111 return mtk::kZero;
00112 }
00113
00114 mtk::Real EastBC(const mtk::Real &yy, const mtk::Real &tt) {
00115
00116 return yy*exp(-0.5*(mtk::kOne + yy*yy));
00117 }
00118
00119 mtk::Real SouthBC(const mtk::Real &xx, const mtk::Real &tt) {
00120
00121 return mtk::kZero;
00122 }
00123
00124 mtk::Real NorthBC(const mtk::Real &xx, const mtk::Real &tt) {
00125
00126 return xx*exp(-0.5*(xx*xx + mtk::kOne));
00127 }
00128
00129 mtk::Real KnownSolution(const mtk::Real &xx, const mtk::Real &yy) {
00130
00131 mtk::Real x_squared{xx*xx};
00132 mtk::Real y_squared{yy*yy};
00133 mtk::Real aux{-0.5*(x_squared + y_squared)};
00134
00135 return xx*yy*exp(aux);
00136 }
00137
00138 int main () {
00139
00140 std::cout << "Example: Poisson Equation on a 2D Uniform Staggered Grid ";
00141 std::cout << "with Dirichlet and Neumann BCs." << std::endl;
00142
00143
00144 mtk::Real west_bndy_x{0.0};
00145 mtk::Real east_bndy_x{1.0};
00146 mtk::Real south_bndy_y{0.0};
00147 mtk::Real north_bndy_y{1.0};
00148 int num_cells_x{5};
00149 int num_cells_y{5};

```

```

00150
00151 mtk::UniStgGrid2D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00152 south_bndy_y, north_bndy_y, num_cells_y);
00153
00154 mtk::Lap2D lap;
00155
00156 if (!lap.ConstructLap2D(comp_sol)) {
00157 std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00158 return EXIT_FAILURE;
00159 }
00160
00161 mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix());
00162
00163 mtk::UniStgGrid2D source(west_bndy_x, east_bndy_x, num_cells_x,
00164 south_bndy_y, north_bndy_y, num_cells_y);
00165
00166 source.BindScalarField(Source);
00167
00168 mtk::RobinBCDescriptor2D bcd;
00169
00170 bcd.PushBackWestCoeff(BCCoeff);
00171 bcd.PushBackEastCoeff(BCCoeff);
00172 bcd.PushBackSouthCoeff(BCCoeff);
00173 bcd.PushBackNorthCoeff(BCCoeff);
00174
00175 bcd.ImposeOnLaplacianMatrix(lap, comp_sol, lapm);
00176
00177 if (!lapm.WriteToFile("poisson_2d_lapm.dat")) {
00178 std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00179 return EXIT_FAILURE;
00180 }
00181
00182 bcd.set_west_condition(WestBC);
00183 bcd.set_east_condition(EastBC);
00184 bcd.set_south_condition(SouthBC);
00185 bcd.set_north_condition(NorthBC);
00186
00187 bcd.ImposeOnGrid(source);
00188
00189 if (!source.WriteToFile("poisson_2d_source.dat", "x", "y", "s(x,y)")) {
00190 std::cerr << "Source term could not be written to disk." << std::endl;
00191 return EXIT_FAILURE;
00192 }
00193
00194 int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00195
00196 if (!info) {
00197 std::cout << "System solved." << std::endl;
00198 std::cout << std::endl;
00199 } else {
00200 std::cerr << "Something wrong solving system! info = " << info << std::endl;
00201 std::cerr << "Exiting..." << std::endl;
00202 return EXIT_FAILURE;
00203 }
00204
00205 if (!source.WriteToFile("poisson_2d_comp_sol.dat", "x", "y", "~u(x,y)")) {
00206 std::cerr << "Solution could not be written to file." << std::endl;
00207 return EXIT_FAILURE;
00208 }
00209
00210 mtk::UniStgGrid2D known_sol(west_bndy_x, east_bndy_x, num_cells_x,
00211 south_bndy_y, north_bndy_y, num_cells_y);
00212
00213 known_sol.BindScalarField(KnownSolution);
00214
00215 if (!known_sol.WriteToFile("poisson_2d_known_sol.dat", "x", "y", "u(x,y)")) {
00216 std::cerr << "Known solution could not be written to file." << std::endl;
00217 return EXIT_FAILURE;
00218 }
00219
00220 mtk::Real relative_norm_2_error{};
00221
00222 relative_norm_2_error =
00223 mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00224 known_sol.discrete_field(),
00225 known_sol.Size());
00226
00227 std::cout << "relative_norm_2_error = ";
00228 std::cout << relative_norm_2_error << std::endl;
00229 }
00230
00231

```



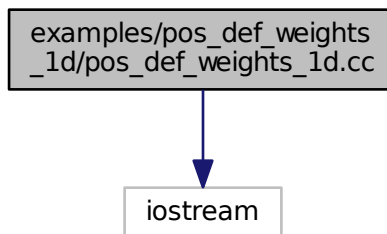
```
00237 #else
00238 #include <iostream>
00239 using std::cout;
00240 using std::endl;
00241 int main () {
00242 cout << "This code HAS to be compiled with support for C++11." << endl;
00243 cout << "Exiting..." << endl;
00244 return EXIT_SUCCESS;
00245 }
00246 #endif
```

## 18.15 examples/pos\_def\_weights\_1d/pos\_def\_weights\_1d.cc File Reference

The CBS algorithm computes positive-definite weights, for 1D operators.

```
#include <iostream>
```

Include dependency graph for pos\_def\_weights\_1d.cc:



### Functions

- int [main](#) ()

#### 18.15.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [pos\\_def\\_weights\\_1d.cc](#).

#### 18.15.2 Function Documentation

##### 18.15.2.1 int main ( )

Definition at line [118](#) of file [pos\\_def\\_weights\\_1d.cc](#).

## 18.16 pos\_def\_weights\_1d.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <vector>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066 std::cout << "Example: Positive-Definite Weights for 1D Mimetic"
00067 "Operators." << std::endl;
00068
00069
00070
00071 mtk::Grad1D grad10;
00072
00073 bool assertion = grad10.ConstructGrad1D(10);
00074 if (!assertion) {
00075 std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00076 return EXIT_FAILURE;
00077 }
00078
00079 mtk::Grad1D grad12;
00080
00081 assertion = grad12.ConstructGrad1D(12);
00082 if (!assertion) {
00083 std::cerr << "Mimetic grad (12th order) could not be built." << std::endl;
00084 return EXIT_FAILURE;
00085 }

```

```

00086
00088
00089 mtk::Div1D div8;
00090
00091 assertion = div8.ConstructDiv1D(8);
00092 if (!assertion) {
00093 std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00094 return EXIT_FAILURE;
00095 }
00096
00097 mtk::Div1D div10;
00098
00099 assertion = div10.ConstructDiv1D(10);
00100 if (!assertion) {
00101 std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00102 return EXIT_FAILURE;
00103 }
00104
00105 mtk::Div1D div12;
00106
00107 assertion = div12.ConstructDiv1D(12);
00108 if (!assertion) {
00109 std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00110 return EXIT_FAILURE;
00111 }
00112 }
00113
00114 #else
00115 #include <iostream>
00116 using std::cout;
00117 using std::endl;
00118 int main () {
00119 cout << "This code HAS to be compiled with support for C++11." << endl;
00120 cout << "Exiting..." << endl;
00121 return EXIT_SUCCESS;
00122 }
00123 #endif

```

## 18.17 include/mtk.h File Reference

Includes the entire API.

```

#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_matrix.h"
#include "mtk_dense_matrix.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
#include "mtk_robin_bc_descriptor_1d.h"
#include "mtk_quad_1d.h"
#include "mtk_interp_1d.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"
#include "mtk_lap_2d.h"
#include "mtk_robin_bc_descriptor_2d.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"
#include "mtk_robin_bc_descriptor_3d.h"

```

Include dependency graph for mtk.h:



### 18.17.1 Detailed Description

This file contains every required header file, thus containing the entire API. In this way, client codes only have to instruct `#include "mtk.h"`.

#### Warning

It is extremely important that the headers are added to this file in a specific order; that is, considering the dependence between the classes these contain.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk.h](#).

## 18.18 mtk.h

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00277 #ifndef MTK_INCLUDE_MTK_H_
00278 #define MTK_INCLUDE_MTK_H_
00279
00287 #include "mtk_roots.h"
00288
00296 #include "mtk_enums.h"
00297
00305 #include "mtk_tools.h"
00306
00314 #include "mtk_matrix.h"
00315 #include "mtk_dense_matrix.h"
00316
00324 #include "mtk_blas_adapter.h"
00325 #include "mtk_lapack_adapter.h"
00326 #include "mtk_glpk_adapter.h"
00327
00335 #include "mtk_uni_stg_grid_1d.h"
00336 #include "mtk_uni_stg_grid_2d.h"
00337 #include "mtk_uni_stg_grid_3d.h"
00338
00346 #include "mtk_grad_1d.h"
00347 #include "mtk_div_1d.h"
00348 #include "mtk_lap_1d.h"
00349 #include "mtk_robin_bc_descriptor_1d.h"
00350 #include "mtk_quad_1d.h"
00351 #include "mtk_interp_1d.h"
00352
00353 #include "mtk_grad_2d.h"
00354 #include "mtk_div_2d.h"
00355 #include "mtk_curl_2d.h"
00356 #include "mtk_lap_2d.h"
00357 #include "mtk_robin_bc_descriptor_2d.h"
```

```

00358
00359 #include "mtk_grad_3d.h"
00360 #include "mtk_div_3d.h"
00361 #include "mtk_lap_3d.h"
00362 #include "mtk_robin_bc_descriptor_3d.h"
00363
00364 #endif // End of: MTK_INCLUDE_MTK_H_

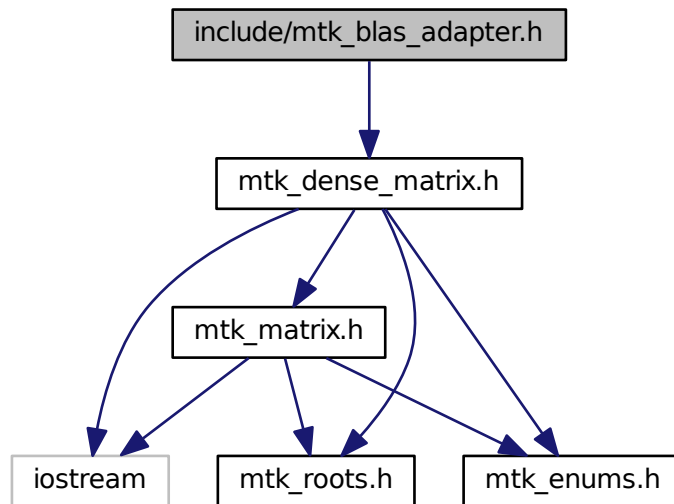
```

## 18.19 include/mtk\_blas\_adapter.h File Reference

Adapter class for the BLAS API.

```
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk\_blas\_adapter.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::BLASAdapter`

*Adapter class for the BLAS API.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.19.1 Detailed Description

Definition of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the BLAS.

The **BLAS (Basic Linear Algebra Subprograms)** are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.

The BLAS can be installed from links given in the See Also section of this page.

#### See also

<http://www.netlib.org/blas/>  
<https://software.intel.com/en-us/non-commercial-software-development>

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_blas\\_adapter.h](#).

## 18.20 mtk\_blas\_adapter.h

```
00001
00025 /*
00026 Copyright (C) 2015, Computational Science Research Center, San Diego State
00027 University. All rights reserved.
00028
00029 Redistribution and use in source and binary forms, with or without modification,
00030 are permitted provided that the following conditions are met:
00031
00032 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00033 and a copy of the modified files should be reported once modifications are
00034 completed, unless these modifications are made through the project's GitHub
00035 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00036 should be developed and included in any deliverable.
00037
00038 2. Redistributions of source code must be done through direct
00039 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00040
00041 3. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 4. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders, and due credit should
00047 be given to the copyright holders.
00048
00049 5. Neither the name of the copyright holder nor the names of its contributors
00050 may be used to endorse or promote products derived from this software without
00051 specific prior written permission.
00052
00053 The copyright holders provide no reassurances that the source code provided does
00054 not infringe any patent, copyright, or any other intellectual property rights of
00055 third parties. The copyright holders disclaim any liability to any recipient for
00056 claims brought against recipient by any third party for infringement of that
00057 parties intellectual property rights.
```

```

00058
00059 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00060 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00061 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00062 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00063 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00064 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00065 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00066 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00067 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00068 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00069 */
00070
00071 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00072 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00073
00074 #include "mtk_dense_matrix.h"
00075
00076 namespace mtk {
00077
00099 class BLASAdapter {
00100 public:
00109 static Real RealNRM2(Real *in, int &in_length);
00110
00127 static void RealAXPY(Real alpha, Real *xx, Real *yy, int &in_length);
00128
00143 static Real RelNorm2Error(Real *computed, Real *known, int length);
00144
00162 static void RealDenseMV(Real &alpha,
00163 DenseMatrix &aa,
00164 Real *xx,
00165 Real &beta,
00166 Real *yy);
00167
00182 static DenseMatrix RealDenseMM(DenseMatrix &aa,
DenseMatrix &bb);
00183
00198 static DenseMatrix RealDenseSM(Real alpha,
DenseMatrix &aa);
00199 };
00200 }
00201 #endif // End of: MTK_INCLUDE_BLAS_ADAPTER_H_

```

## 18.21 include/mtk\_curl\_2d.h File Reference

Includes the definition of the class Curl2D.

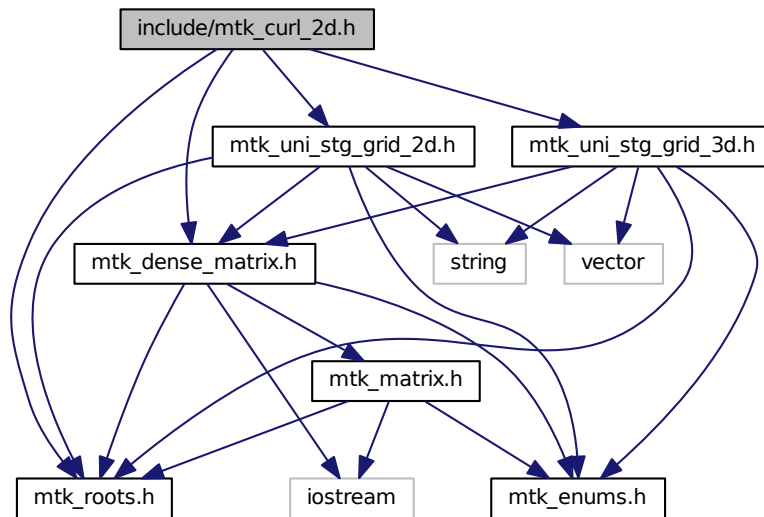
```

#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"

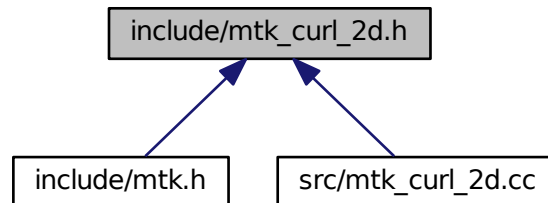
```



Include dependency graph for mtk\_curl\_2d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Curl2D](#)  
*Implements a 2D mimetic curl operator.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.21.1 Detailed Description

This class implements a 2D curl operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_curl\\_2d.h](#).

## 18.22 mtk\_curl\_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_CURL_2D_H_
00058 #define MTK_INCLUDE_MTK_CURL_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk{
00066
00077 class Curl2D {
00078 public:
00080 UniStgGrid3D operator*(const UniStgGrid2D &grid) const;
00081

```

```

00083 Curl2D();
00084
00090 Curl2D(const Curl2D &curl);
00091
00093 ~Curl2D();
00094
00100 bool ConstructCurl2D(const UniStgGrid2D &grid,
00101 int order_accuracy = kDefaultOrderAccuracy,
00102 Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109 DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112 DenseMatrix curl_;
00113
00114 int order_accuracy_;
00115
00116 Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_CURL_2D_H_

```

## 18.23 include/mtk\_dense\_matrix.h File Reference

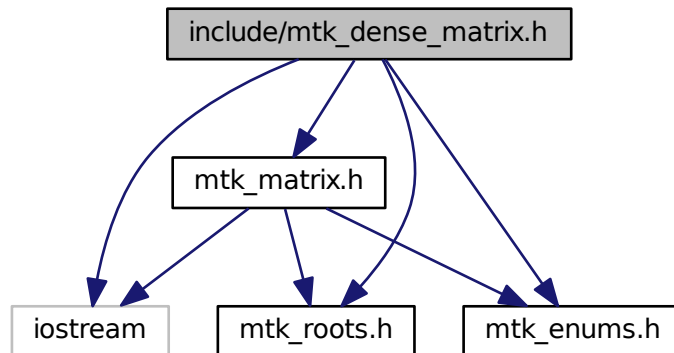
Defines a common dense matrix, using a 1D array.

```

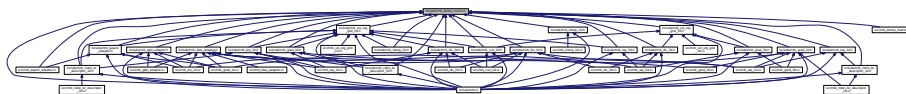
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_matrix.h"

```

Include dependency graph for mtk\_dense\_matrix.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::DenseMatrix](#)

*Defines a common dense matrix, using a 1D array.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.23.1 Detailed Description

For developing purposes, it is better to have a not-so-intricated data structure implementing matrices. This is the purpose of this class: to be used for prototypes of new code for small test cases. In every other instance, this should be replaced by the most appropriate sparse matrix.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

#### Note

We prefer composition to inheritance [Reedy, 2011]. The main reason for this preference is that inheritance produces a more tightly coupled design. When a class inherits from another type be it public, protected, or private inheritance the subclass gains access to all public and protected members of the base class, whereas with composition, the class is only coupled to the public members of the other class. Furthermore, if you only hold a pointer to the other object, then your interface can use a forward declaration of the class rather than `#include` its full definition. This results in greater compile-time insulation and improves the time it takes to compile your code.

Definition in file [mtk\\_dense\\_matrix.h](#).

## 18.24 mtk\_dense\_matrix.h

```

00001
00023 /*
00024 Copyright (C) 2015, Computational Science Research Center, San Diego State
00025 University. All rights reserved.
00026
00027 Redistribution and use in source and binary forms, with or without modification,
00028 are permitted provided that the following conditions are met:
00029
00030 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00031 and a copy of the modified files should be reported once modifications are
00032 completed, unless these modifications are made through the project's GitHub
00033 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00034 should be developed and included in any deliverable.
00035
00036 2. Redistributions of source code must be done through direct
00037 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00038
00039 3. Redistributions in binary form must reproduce the above copyright notice,
00040 this list of conditions and the following disclaimer in the documentation and/or
00041 other materials provided with the distribution.
00042
00043 4. Usage of the binary form on proprietary applications shall require explicit
00044 prior written permission from the the copyright holders, and due credit should
00045 be given to the copyright holders.
00046
00047 5. Neither the name of the copyright holder nor the names of its contributors

```

```

00048 may be used to endorse or promote products derived from this software without
00049 specific prior written permission.
00050
00051 The copyright holders provide no reassurances that the source code provided does
00052 not infringe any patent, copyright, or any other intellectual property rights of
00053 third parties. The copyright holders disclaim any liability to any recipient for
00054 claims brought against recipient by any third party for infringement of that
00055 parties intellectual property rights.
00056
00057 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00058 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00059 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00060 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00061 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00062 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00063 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00064 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00065 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00066 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00067 */
00068
00069 #ifndef MTK_INCLUDE_DENSE_MATRIX_H_
00070 #define MTK_INCLUDE_DENSE_MATRIX_H_
00071
00072 #include <iostream>
00073
00074 #include "mtk_roots.h"
00075 #include "mtk_enums.h"
00076 #include "mtk_matrix.h"
00077
00078 namespace mtk {
00079
00092 class DenseMatrix {
00093 public:
00095 friend std::ostream& operator <<(std::ostream &stream, DenseMatrix &in);
00096
00104 DenseMatrix& operator =(const DenseMatrix &in);
00105
00107 bool operator ==(const DenseMatrix &in);
00108
00110 DenseMatrix();
00111
00117 DenseMatrix(const DenseMatrix &in);
00118
00127 DenseMatrix(const int &num_rows, const int &num_cols);
00128
00154 DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00155
00189 DenseMatrix(const Real *const gen,
00190 const int &gen_length,
00191 const int &pro_length,
00192 const bool &transpose);
00193
00195 ~DenseMatrix();
00196
00202 Matrix matrix_properties() const noexcept;
00203
00209 int num_rows() const noexcept;
00210
00216 int num_cols() const noexcept;
00217
00223 Real* data() const noexcept;
00224
00232 void SetOrdering(mtk::MatrixOrdering oo) noexcept;
00233
00242 Real GetValue(const int &row_coord, const int &col_coord) const noexcept;
00243
00251 void SetValue(const int &row_coord,
00252 const int &col_coord,
00253 const Real &val) noexcept;
00254
00256 void Transpose();
00257
00259 void OrderRowMajor();
00260
00262 void OrderColMajor();
00263
00274 static DenseMatrix Kron(const DenseMatrix &aa,
00275 const DenseMatrix &bb);
00276
00286 bool WriteToFile(const std::string &filename) const;

```

```

00287
00288 private:
00289 Matrix matrix_properties;
00290
00291 Real *data_;
00292 };
00293 }
00294 #endif // End of: MTK_INCLUDE_MTK_DENSE_MATRIX_H_

```

## 18.25 include/mtk\_div\_1d.h File Reference

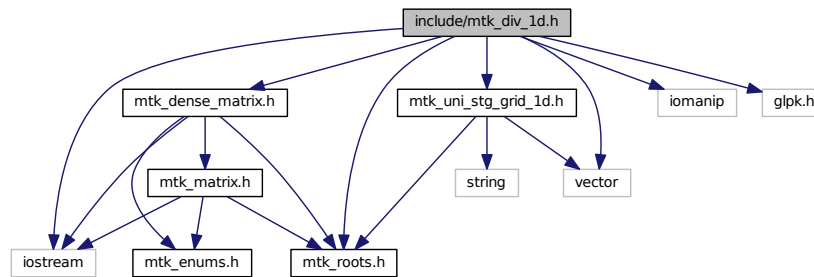
Includes the definition of the class Div1D.

```

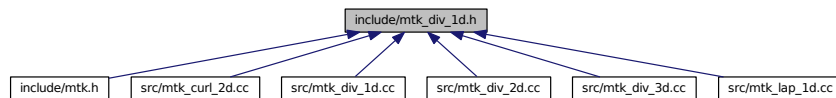
#include <iostream>
#include <iomanip>
#include <vector>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk\_div\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Div1D](#)  
*Implements a 1D mimetic divergence operator.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.25.1 Detailed Description

This class implements a 1D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_1d.h](#).

## 18.26 mtk\_div\_1d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_DIV_1D_H_
00058 #define MTK_INCLUDE_DIV_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include <vector>
00064

```

```

00065 #include "glpk.h"
00066
00067 #include "mtk_roots.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_id.h"
00070
00071 namespace mtk {
00072
00083 class Div1D {
00084 public:
00086 friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00087
00089 Div1D();
00090
00096 Div1D(const Div1D &div);
00097
00099 ~Div1D();
00100
00106 bool ConstructDiv1D(int order_accuracy = kDefaultOrderAccuracy,
00107 Real mimetic_threshold = kDefaultMimeticThreshold);
00108
00114 int num_bndy_coeffs() const;
00115
00121 Real *coeffs_interior() const;
00122
00128 Real *weights_crs(void) const;
00129
00135 Real *weights_cbs(void) const;
00136
00142 DenseMatrix mim_bndy() const;
00143
00149 DenseMatrix ReturnAsDenseMatrix(const
00150 UniStgGrid1D &grid) const;
00156 DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
00157 const;
00158 private:
00164 bool ComputeStencilInteriorGrid(void);
00165
00172 bool ComputeRationalBasisNullSpace(void);
00173
00179 bool ComputePreliminaryApproximations(void);
00180
00186 bool ComputeWeights(void);
00187
00193 bool ComputeStencilBoundaryGrid(void);
00194
00200 bool AssembleOperator(void);
00201
00202 int order_accuracy_;
00203 int dim_null_;
00204 int num_bndy_coeffs_;
00205 int divergence_length_;
00206 int minrow_;
00207 int row_;
00208
00209 DenseMatrix rat_basis_null_space_;
00210
00211 Real *coeffs_interior_;
00212 Real *prem_apps_;
00213 Real *weights_crs_;
00214 Real *weights_cbs_;
00215 Real *mim_bndy_;
00216 Real *divergence_;
00217
00218 std::vector<Real> sum_rows_mim_bndy_;
00219
00220 Real mimetic_threshold_;
00221 };
00222 }
00223 #endif // End of: MTK_INCLUDE_DIV_1D_H_

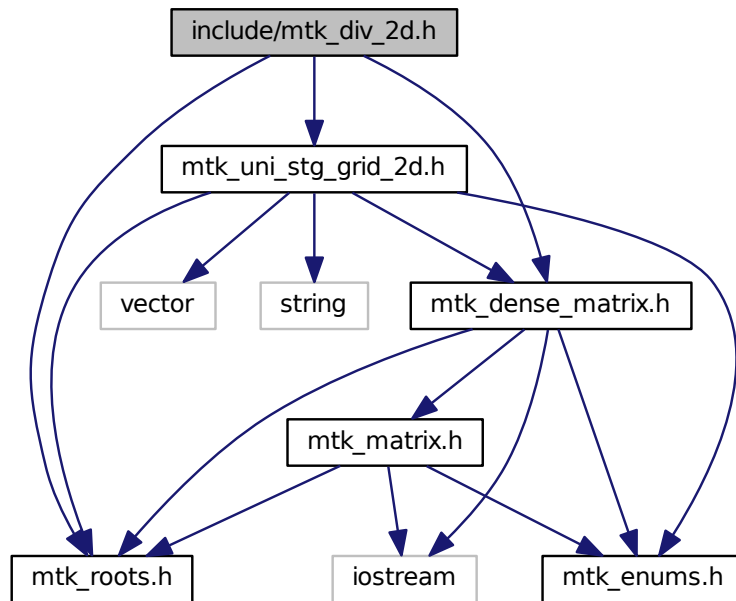
```

## 18.27 include/mtk\_div\_2d.h File Reference

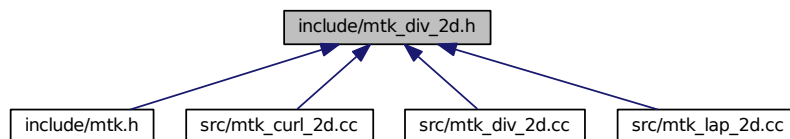
Includes the definition of the class Div2D.



```
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_div_2d.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Div2D](#)  
*Implements a 2D mimetic divergence operator.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.27.1 Detailed Description

This class implements a 2D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_2d.h](#).

## 18.28 mtk\_div\_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_2D_H_
00058 #define MTK_INCLUDE_MTK_DIV_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{

```

```

00065
00076 class Div2D {
00077 public:
00079 Div2D();
00080
00086 Div2D(const Div2D &div);
00087
00089 ~Div2D();
00090
00096 bool ConstructDiv2D(const UniStgGrid2D &grid,
00097 int order_accuracy = kDefaultOrderAccuracy,
00098 Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105 DenseMatrix ReturnAsDenseMatrix() const;
00106
00107 private:
00108 DenseMatrix divergence_;
00109
00110 int order_accuracy_;
00111
00112 Real mimetic_threshold_;
00113 };
00114
00115 #endif // End of: MTK_INCLUDE_MTK_DIV_2D_H_

```

## 18.29 include/mtk\_div\_3d.h File Reference

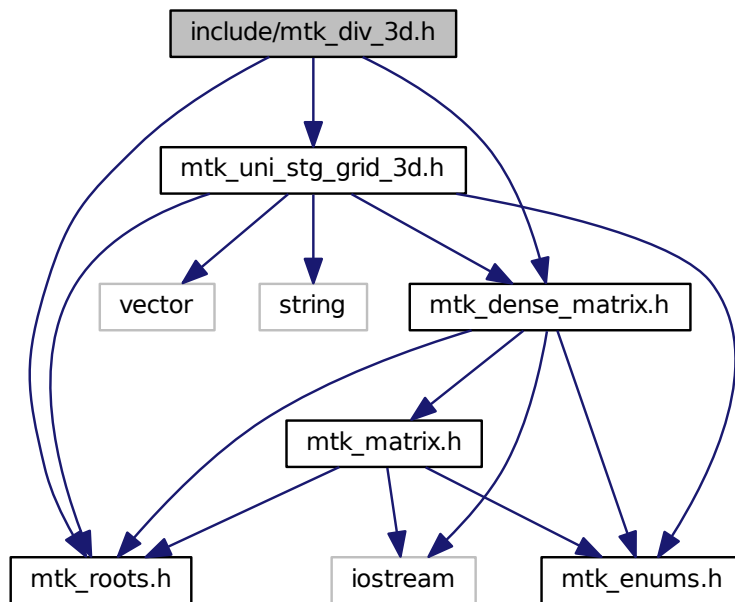
Includes the definition of the class Div3D.

```

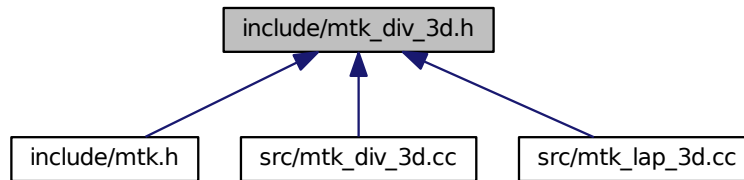
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for mtk\_div\_3d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::Div3D`  
*Implements a 3D mimetic divergence operator.*

## Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

### 18.29.1 Detailed Description

This class implements a 3D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_3d.h](#).

### 18.30 mtk\_div\_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026

```

```

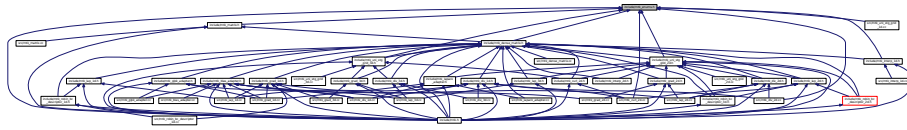
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_3D_H_
00058 #define MTK_INCLUDE_MTK_DIV_3D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{
00065
00066 class Div3D {
00067 public:
00068 Div3D();
00069
00070 Div3D(const Div3D &div);
00071
00072 ~Div3D();
00073
00074 bool ConstructDiv3D(const UniStgGrid3D &grid,
00075 int order_accuracy = kDefaultOrderAccuracy,
00076 Real mimetic_threshold = kDefaultMimeticThreshold);
00077
00078 DenseMatrix ReturnAsDenseMatrix() const;
00079
00080 private:
00081 DenseMatrix divergence_;
00082
00083 int order_accuracy_;
00084
00085 Real mimetic_threshold_;
00086 };
00087
00088 #endif // End of: MTK_INCLUDE_MTK_DIV_3D_H_

```

## 18.31 include/mtk\_enums.h File Reference

Considered enumeration types in the MTK.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Enumerations

- enum [mtk::MatrixStorage](#) { [mtk::MatrixStorage::DENSE](#), [mtk::MatrixStorage::BANDED](#), [mtk::MatrixStorage::CRS](#) }  
*Considered matrix storage schemes to implement sparse matrices.*
- enum [mtk::MatrixOrdering](#) { [mtk::MatrixOrdering::ROW\\_MAJOR](#), [mtk::MatrixOrdering::COL\\_MAJOR](#) }  
*Considered matrix ordering (for Fortran purposes).*
- enum [mtk::FieldNature](#) { [mtk::FieldNature::SCALAR](#), [mtk::FieldNature::VECTOR](#) }  
*Nature of the field discretized in a given grid.*
- enum [mtk::DirInterp](#) { [mtk::DirInterp::SCALAR\\_TO\\_VECTOR](#), [mtk::DirInterp::VECTOR\\_TO\\_SCALAR](#) }  
*Interpolation operator.*

### 18.31.1 Detailed Description

Enumeration types are used throughout the MTK to differentiate instances of derived classes, as well as for mnemonic purposes. In this file, the enumeration types are listed alphabetically.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_enums.h](#).

## 18.32 mtk\_enums.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct

```

```

00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_ENUMS_H_
00059 #define MTK_INCLUDE_ENUMS_H_
00060
00061 namespace mtk {
00062
00063 enum class MatrixStorage {
00064 DENSE,
00065 BANDED,
00066 CRS
00067 };
00068
00069 enum class MatrixOrdering {
00070 ROW_MAJOR,
00071 COL_MAJOR
00072 };
00073
00074 enum class FieldNature {
00075 SCALAR,
00076 VECTOR
00077 };
00078
00079 enum class DirInterp {
00080 SCALAR_TO_VECTOR,
00081 VECTOR_TO_SCALAR
00082 };
00083
00084 #endif // End of: MTK_INCLUDE_ENUMS_H_

```

## 18.33 include/mtk\_glpk\_adapter.h File Reference

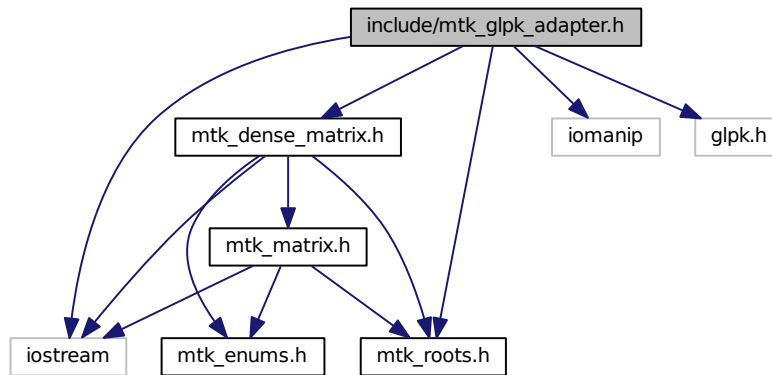
Adapter class for the GLPK API.

```

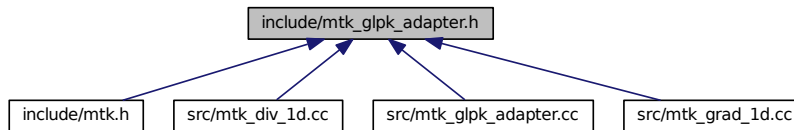
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"

```

Include dependency graph for `mtk_glpk_adapter.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::GLPKAdapter](#)  
*Adapter class for the GLPK API.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.33.1 Detailed Description

Definition of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the GLPK.

The **GLPK (GNU Linear Programming Kit)** package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.



See also

<http://www.gnu.org/software/glpk/>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_glpk\\_adapter.h](#).

## 18.34 mtk\_glpk\_adapter.h

```

00001
00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00067 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00068
00069 #include <iostream>
00070 #include <iomanip>
00071
00072 #include "glpk.h"
00073
00074 #include "mtk_roots.h"
00075 #include "mtk_dense_matrix.h"
00076
00077 namespace mtk {
00078
00102 class GLPKAdapter {

```

```

00103 public:
00124 static mtk::Real SolveSimplexAndCompare(
00125 mtk::Real *A,
00126 int nrows,
00127 int ncols,
00128 int kk,
00129 mtk::Real *hh,
00130 mtk::Real *qq,
00131 int robjective,
00132 mtk::Real mimetic_tol,
00133 int copy);
00134 }
00135 #endif // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_

```

## 18.35 include/mtk\_grad\_1d.h File Reference

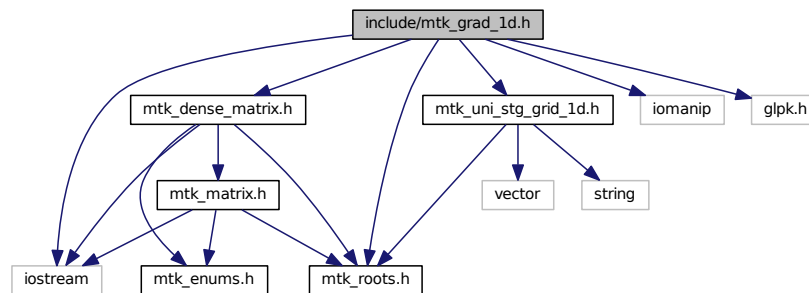
Includes the definition of the class Grad1D.

```

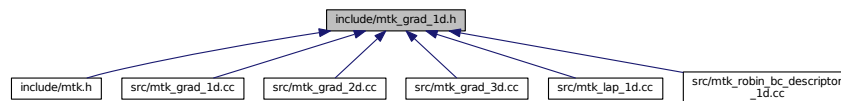
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk\_grad\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Grad1D](#)

*Implements a 1D mimetic gradient operator.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.35.1 Detailed Description

This class implements a 1D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_1d.h](#).

## 18.36 mtk\_grad\_1d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_GRAD_1D_H_
00058 #define MTK_INCLUDE_GRAD_1D_H_
00059
00060 #include <iostream>

```

```

00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Grad1D {
00082 public:
00084 friend std::ostream& operator <<(std::ostream& stream, Grad1D &in);
00085
00087 Grad1D();
00088
00094 Grad1D(const Grad1D &grad);
00095
00097 ~Grad1D();
00098
00104 bool ConstructGrad1D(int order_accuracy = kDefaultOrderAccuracy,
00105 Real mimetic_threshold = kDefaultMimeticThreshold);
00106
00112 int num_bndy_coeffs() const;
00113
00119 Real *coeffs_interior() const;
00120
00126 Real *weights_crs(void) const;
00127
00133 Real *weights_cbs(void) const;
00134
00140 DenseMatrix mim_bndy() const;
00141
00147 DenseMatrix ReturnAsDenseMatrix(Real west,
00148 Real east, int num_cells_x) const;
00154 DenseMatrix ReturnAsDenseMatrix(const
00155 UniStgGrid1D &grid) const;
00161 DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
00162 const;
00163 private:
00169 bool ComputeStencilInteriorGrid(void);
00170
00177 bool ComputeRationalBasisNullSpace(void);
00178
00184 bool ComputePreliminaryApproximations(void);
00185
00191 bool ComputeWeights(void);
00192
00198 bool ComputeStencilBoundaryGrid(void);
00199
00205 bool AssembleOperator(void);
00206
00207 int order_accuracy_;
00208 int dim_null_;
00209 int num_bndy_approxs_;
00210 int num_bndy_coeffs_;
00211 int gradient_length_;
00212 int minrow_;
00213 int row_;
00214
00215 DenseMatrix rat_basis_null_space_;
00216
00217 Real *coeffs_interior_;
00218 Real *prem_apps_;
00219 Real *weights_crs_;
00220 Real *weights_cbs_;
00221 Real *mim_bndy_;
00222 Real *gradient_;
00223
00224 Real mimetic_threshold_;
00225 };
00226 }
00227 #endif // End of: MTK_INCLUDE_GRAD_1D_H_

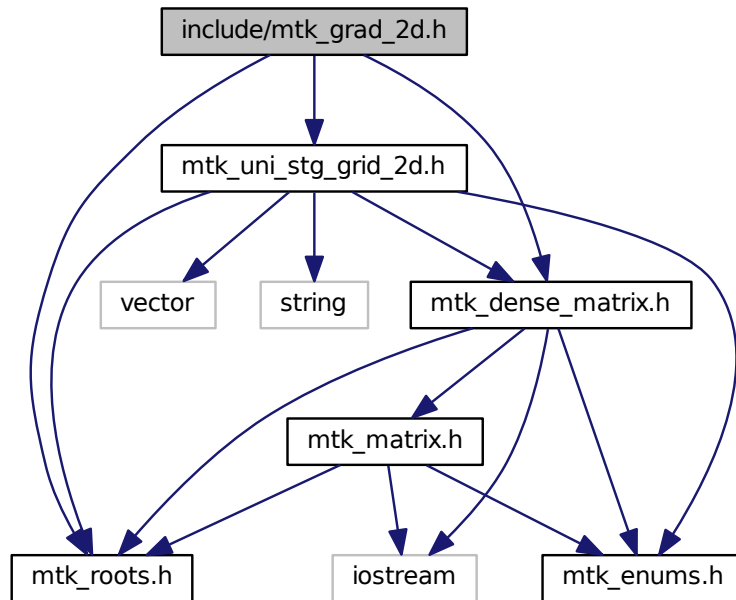
```

## 18.37 include/mtk\_grad\_2d.h File Reference

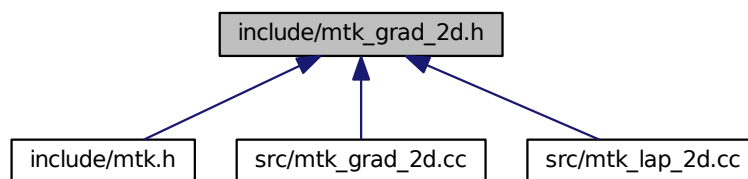
Includes the definition of the class Grad2D.

```
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk\_grad\_2d.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [mtk::Grad2D](#)

*Implements a 2D mimetic gradient operator.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.37.1 Detailed Description

This class implements a 2D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↵BSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_2d.h](#).

## 18.38 mtk\_grad\_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */

```

```

00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_2D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Grad2D {
00077 public:
00078 Grad2D();
00080
00086 Grad2D(const Grad2D &grad);
00087
00089 ~Grad2D();
00090
00096 bool ConstructGrad2D(const UniStgGrid2D &grid,
00097 int order_accuracy = kDefaultOrderAccuracy,
00098 Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105 DenseMatrix ReturnAsDenseMatrix() const;
00106
00107 private:
00108 DenseMatrix gradient_;
00109
00110 int order_accuracy_;
00111
00112 Real mimetic_threshold_;
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_GRAD_2D_H_

```

## 18.39 include/mtk\_grad\_3d.h File Reference

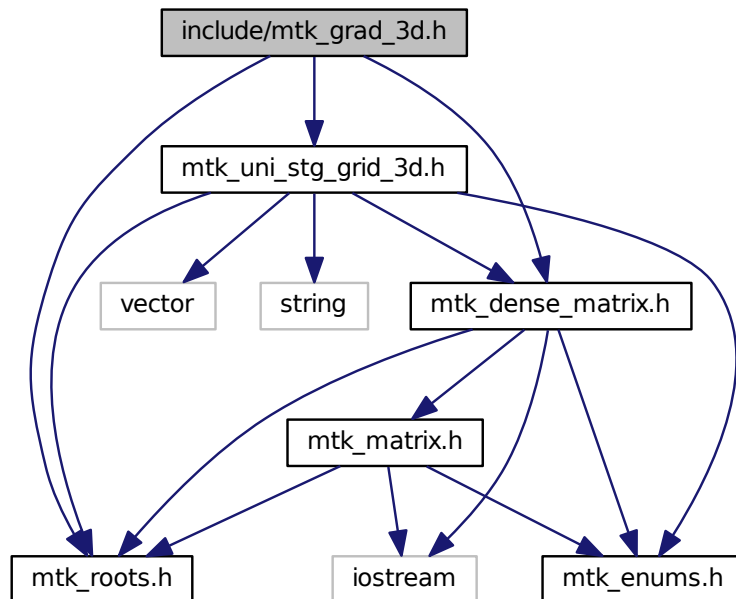
Includes the definition of the class Grad3D.

```

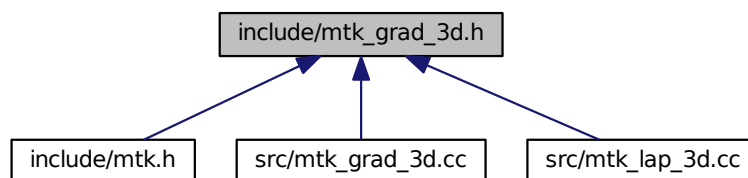
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for `mtk_grad_3d.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Grad3D](#)  
*Implements a 3D mimetic gradient operator.*

## Namespaces

- [mtk](#)



*Mimetic Methods Toolkit namespace.*

### 18.39.1 Detailed Description

This class implements a 3D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_3d.h](#).

## 18.40 mtk\_grad\_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_3D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_3D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{

```

```

00065
00076 class Grad3D {
00077 public:
00079 Grad3D();
00080
00086 Grad3D(const Grad3D &grad);
00087
00089 ~Grad3D();
00090
00096 bool ConstructGrad3D(const UniStgGrid3D &grid,
00097 int order_accuracy = kDefaultOrderAccuracy,
00098 Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105 DenseMatrix ReturnAsDenseMatrix() const;
00106
00107 private:
00108 DenseMatrix gradient_;
00109
00110 int order_accuracy_;
00111
00112 Real mimetic_threshold_;
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_GRAD_3D_H_

```

## 18.41 include/mtk\_interp\_1d.h File Reference

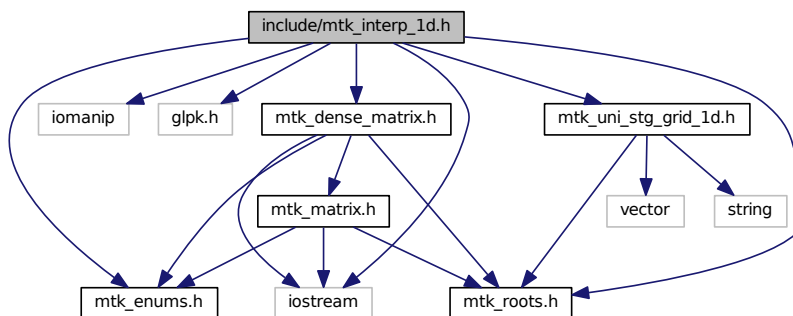
Includes the definition of the class Interp1D.

```

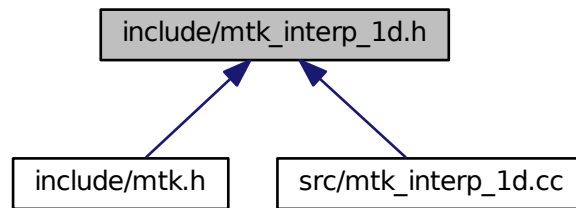
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk\_interp\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::Interp1D`  
*Implements a 1D interpolation operator.*

## Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

### 18.41.1 Detailed Description

This class implements a 1D interpolation operator.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu  
: Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file `mtk_interp_1d.h`.

## 18.42 mtk\_interp\_1d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024

```

```

00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_INTERP_1D_H_
00059 #define MTK_INCLUDE_INTERP_1D_H_
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "glpk.h"
00065
00066 #include "mtk_roots.h"
00067 #include "mtk_enums.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_1d.h"
00070
00071 namespace mtk {
00072
00082 class Interp1D {
00083 public:
00085 friend std::ostream& operator <<(std::ostream& stream, Interp1D &in);
00086
00088 Interp1D();
00089
00095 Interp1D(const Interp1D &interp);
00096
00098 ~Interp1D();
00099
00105 bool ConstructInterp1D(int order_accuracy =
kDefaultOrderAccuracy,
00106 mtk::DirInterp dir = SCALAR_TO_VECTOR);
00107
00113 Real *coeffs_interior() const;
00114
00120 DenseMatrix ReturnAsDenseMatrix(const
UniStgGrid1D &grid) const;
00121
00122 private:
00123 DirInterp dir_interp_;
00124
00125 int order_accuracy_;
00126
00127 Real *coeffs_interior_;
00128 };
00129
00130 #endif // End of: MTK_INCLUDE_INTERP_1D_H_

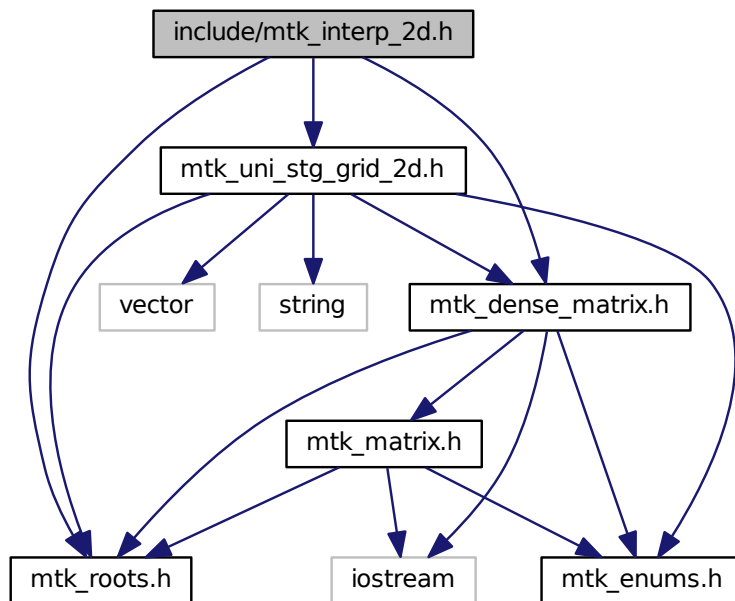
```

## 18.43 include/mtk\_interp\_2d.h File Reference

Includes the definition of the class Interp2D.

```
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk\_interp\_2d.h:



### Classes

- class `mtk::Interp2D`  
*Implements a 2D interpolation operator.*

### Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

#### 18.43.1 Detailed Description

This class implements a 2D interpolation operator.

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu  
 : Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file [mtk\\_interp\\_2d.h](#).

**18.44 mtk\_interp\_2d.h**

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_MTK_INTERP_2D_H_
00059 #define MTK_INCLUDE_MTK_INTERP_2D_H_
00060
00061 #include "mtk_roots.h"
00062 #include "mtk_dense_matrix.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk{
00066
00076 class Interp2D {
00077 public:
00079 Interp2D();
00080
00086 Interp2D(const Interp2D &interp);
00087
00089 ~Interp2D();
00090
00096 DenseMatrix ConstructInterp2D(const UniStgGrid2D &grid,
00097 int order_accuracy = kDefaultOrderAccuracy,

```

```

00098 Real mimetic_threshold =
00099 kDefaultMimeticThreshold);
00105 DenseMatrix ReturnAsDenseMatrix();
00106
00107 private:
00108 DenseMatrix interpolator_;
00109
00110 int order_accuracy_;
00111
00112 Real mimetic_threshold_;
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_INTERP_2D_H_

```

## 18.45 include/mtk\_lap\_1d.h File Reference

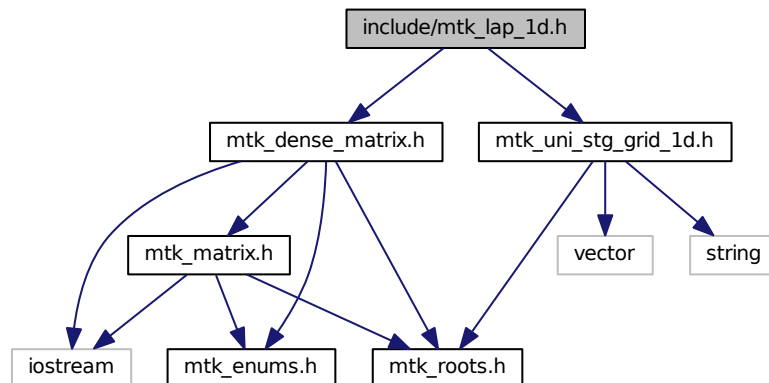
Includes the definition of the class Lap1D.

```

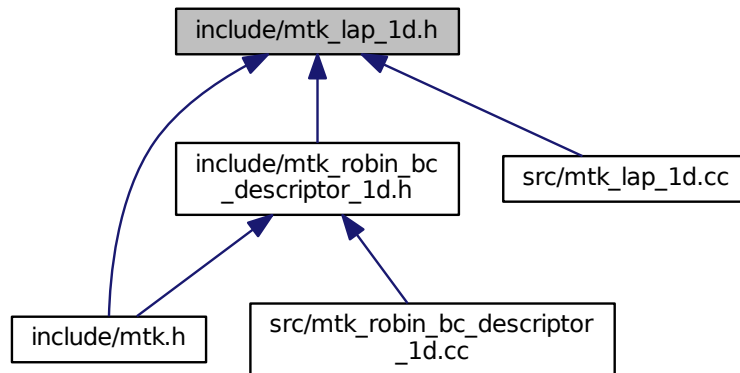
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk\_lap\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Lap1D](#)  
*Implements a 1D mimetic Laplacian operator.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.45.1 Detailed Description

This class implements a 1D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_1d.h](#).

## 18.46 mtk\_lap\_1d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017

```



```

00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_LAP_1D_H_
00058 #define MTK_INCLUDE_LAP_1D_H_
00059
00060 #include "mtk_dense_matrix.h"
00061
00062 #include "mtk_uni_stg_grid_1d.h"
00063
00064 namespace mtk {
00065
00066 class Lap1D {
00067 public:
00068 friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00069
00070 Lap1D();
00071
00072 Lap1D(const Lap1D &lap);
00073
00074 ~Lap1D();
00075
00076 int order_accuracy() const;
00077
00078 Real mimetic_threshold() const;
00079
00080 Real delta() const;
00081
00082 bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00083 Real mimetic_threshold = kDefaultMimeticThreshold);
00084
00085 DenseMatrix ReturnAsDenseMatrix(const
00086 UniStgGrid1D &grid) const;
00087
00088 const mtk::Real* data(const UniStgGrid1D &grid) const;
00089
00090 private:
00091 int order_accuracy_;
00092 int laplacian_length_;
00093
00094 Real *laplacian_;
00095
00096 mutable Real delta_;
00097
00098 Real mimetic_threshold_;

```

```

00146 };
00147 }
00148 #endif // End of: MTK_INCLUDE_LAP_1D_H_

```

## 18.47 include/mtk\_lap\_2d.h File Reference

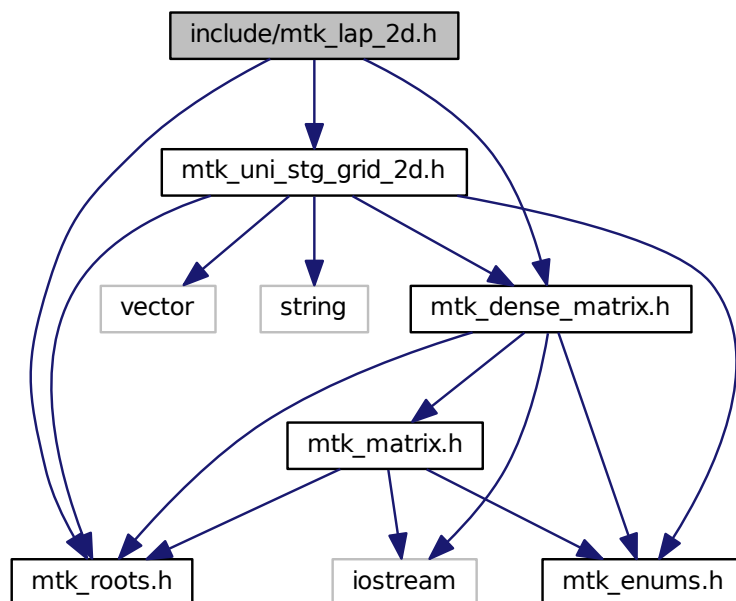
Includes the implementation of the class Lap2D.

```

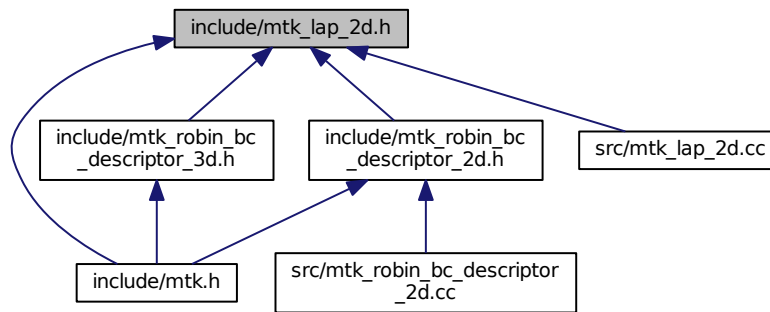
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk\_lap\_2d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::Lap2D`  
*Implements a 2D mimetic Laplacian operator.*

## Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

### 18.47.1 Detailed Description

This class implements a 2D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_2d.h](#).

## 18.48 mtk\_lap\_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications

```

```

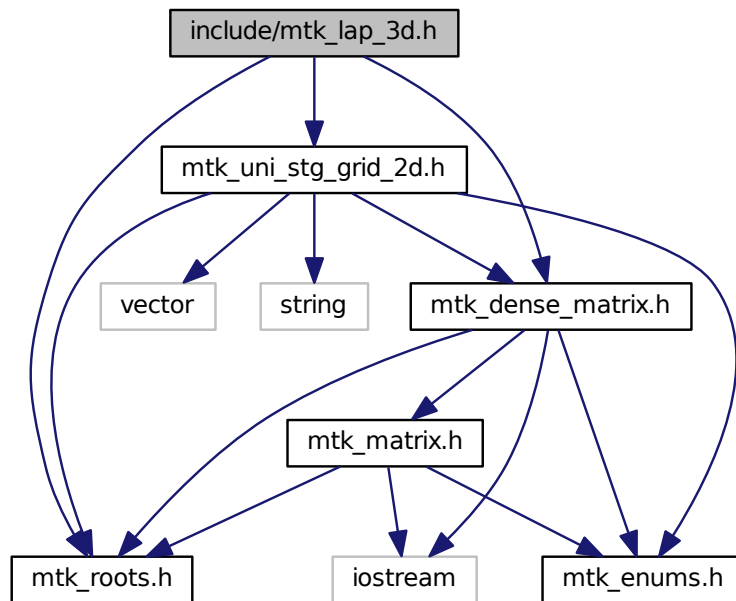
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_2D_H_
00058 #define MTK_INCLUDE_MTK_LAP_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap2D {
00077 public:
00078 Lap2D();
00079
00080 Lap2D(const Lap2D &lap);
00081
00082 ~Lap2D();
00083
00084 bool ConstructLap2D(const UniStgGrid2D &grid,
00085 int order_accuracy = kDefaultOrderAccuracy,
00086 Real mimetic_threshold = kDefaultMimeticThreshold);
00087
00088 DenseMatrix ReturnAsDenseMatrix() const;
00089
00090 Real *data() const;
00091
00092 private:
00093 DenseMatrix laplacian_;
00094
00095 int order_accuracy_;
00096 Real mimetic_threshold_;
00097 };
00098
00099 #endif // End of: MTK_INCLUDE_MTK_LAP_2D_H_

```

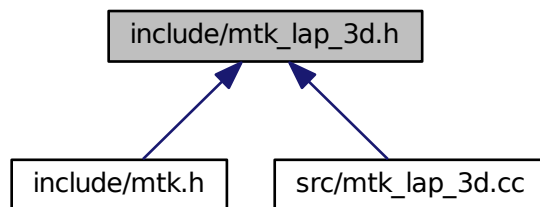
## 18.49 include/mtk\_lap\_3d.h File Reference

Includes the implementation of the class Lap3D.

```
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_lap_3d.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Lap3D](#)

*Implements a 3D mimetic Laplacian operator.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.49.1 Detailed Description

This class implements a 3D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_3d.h](#).

### 18.50 mtk\_lap\_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_3D_H_
00058 #define MTK_INCLUDE_MTK_LAP_3D_H_
00059
00060 #include "mtk_roots.h"

```

```

00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap3D {
00077 public:
00079 UniStgGrid3D operator*(const UniStgGrid3D &grid) const;
00080
00082 Lap3D();
00083
00089 Lap3D(const Lap3D &lap);
00090
00092 ~Lap3D();
00093
00099 bool ConstructLap3D(const UniStgGrid3D &grid,
00100 int order_accuracy = kDefaultOrderAccuracy,
00101 Real mimetic_threshold = kDefaultMimeticThreshold);
00102
00108 DenseMatrix ReturnAsDenseMatrix() const;
00109
00115 Real *data() const;
00116
00117 private:
00118 DenseMatrix laplacian_;
00119
00120 int order_accuracy_;
00121
00122 Real mimetic_threshold_;
00123 };
00124 }
00125 #endif // End of: MTK_INCLUDE_MTK_LAP_3D_H_

```

## 18.51 include/mtk\_lapack\_adapter.h File Reference

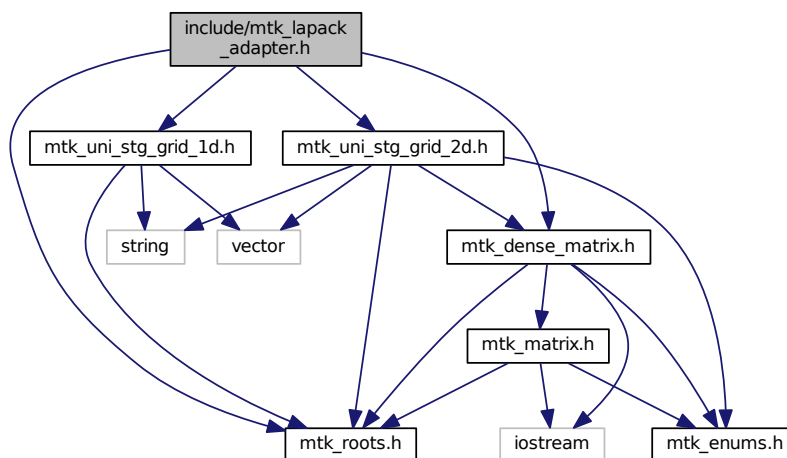
Adapter class for the LAPACK API.

```

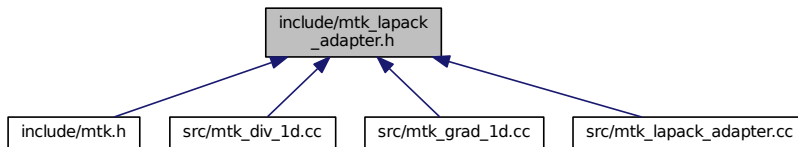
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk\_lapack\_adapter.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::LAPACKAdapter](#)  
*Adapter class for the LAPACK API.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.51.1 Detailed Description

Definition of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the LAPACK.

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

#### See also

<http://www.netlib.org/lapack/>

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lapack\\_adapter.h](#).

## 18.52 mtk\_lapack\_adapter.h

```

00001
00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026

```



```

00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00067 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00068
00069 #include "mtk_roots.h"
00070 #include "mtk_dense_matrix.h"
00071 #include "mtk_uni_stg_grid_ld.h"
00072 #include "mtk_uni_stg_grid_2d.h"
00073
00074 namespace mtk {
00075
00094 class LAPACKAdapter {
00095 public:
00106 static int SolveDenseSystem(mtk::DenseMatrix &mm,
00107 mtk::Real *rhs);
00108
00119 static int SolveDenseSystem(mtk::DenseMatrix &mm,
00120 mtk::DenseMatrix &rr);
00121
00132 static int SolveDenseSystem(mtk::DenseMatrix &mm,
00133 mtk::UniStgGrid1D &rhs);
00134
00146 static int SolveDenseSystem(mtk::DenseMatrix &mm,
00147 mtk::UniStgGrid2D &rhs);
00148
00160 static int SolveRectangularDenseSystem(const
00161 mtk::DenseMatrix &aa,
00162 mtk::Real *ob_,
00163 int ob_ld_);
00175 static mtk::DenseMatrix QRFactorDenseMatrix(
00176 DenseMatrix &matrix);
00176 };
00177 }
00178 #endif // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_

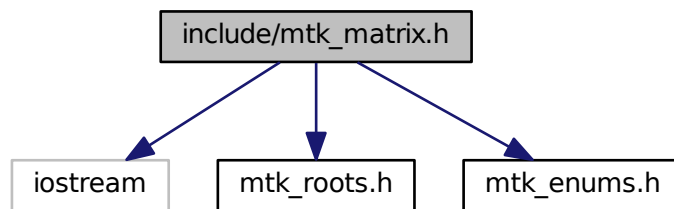
```

## 18.53 include/mtk\_matrix.h File Reference

Definition of the representation of a matrix in the MTK.

```
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
```

Include dependency graph for mtk\_matrix.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [mtk::Matrix](#)

*Definition of the representation of a matrix in the MTK.*

### Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

#### 18.53.1 Detailed Description

Definition of the representation for the matrices implemented in the MTK.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_matrix.h](#).

## 18.54 mtk\_matrix.h

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_MATRIX_H_
00057 #define MTK_INCLUDE_MATRIX_H_
00058
00059 #include <iostream>
00060
00061 #include "mtk_roots.h"
00062 #include "mtk_enums.h"
00063
00064 namespace mtk {
00065
00075 class Matrix {
00076 public:
00077 Matrix();
00078
00079 Matrix(const Matrix &in);
00080
00081 ~Matrix() noexcept ;
00082
00083 MatrixStorage storage() const noexcept;
00084
00085 MatrixOrdering ordering() const noexcept;
00086
00087 int num_rows() const noexcept;
00088
00089 int num_cols() const noexcept;
00090
00091 int num_values() const noexcept;
00092
00093 int ld() const noexcept;
00094
00095 int num_zero() const noexcept;

```

```

00142
00148 int num_non_zero() const noexcept;
00149
00157 int num_null() const noexcept;
00158
00166 int num_non_null() const noexcept;
00167
00173 int kl() const noexcept;
00174
00180 int ku() const noexcept;
00181
00187 int bandwidth() const noexcept;
00188
00196 Real abs_density() const noexcept;
00197
00205 Real rel_density() const noexcept;
00206
00214 Real abs_sparsity() const noexcept;
00215
00223 Real rel_sparsity() const noexcept;
00224
00232 void set_storage(const MatrixStorage &tt) noexcept;
00233
00241 void set_ordering(const MatrixOrdering &oo) noexcept;
00242
00248 void set_num_rows(const int &num_rows) noexcept;
00249
00255 void set_num_cols(const int &num_cols) noexcept;
00256
00262 void set_num_zero(const int &in) noexcept;
00263
00269 void set_num_null(const int &in) noexcept;
00270
00272 void IncreaseNumZero() noexcept;
00273
00275 void IncreaseNumNull() noexcept;
00276
00277 private:
00278 MatrixStorage storage_;
00279
00280 MatrixOrdering ordering_;
00281
00282 int num_rows_;
00283 int num_cols_;
00284 int num_values_;
00285 int ld_;
00286
00287 int num_zero_;
00288 int num_non_zero_;
00289 int num_null_;
00290 int num_non_null_;
00291
00292 int kl_;
00293 int ku_;
00294 int bandwidth_;
00295
00296 Real abs_density_;
00297 Real rel_density_;
00298 Real abs_sparsity_;
00299 Real rel_sparsity_;
00300 };
00301 }
00302 #endif // End of: MTK_INCLUDE_MATRIX_H_

```

## 18.55 include/mtk\_quad\_1d.h File Reference

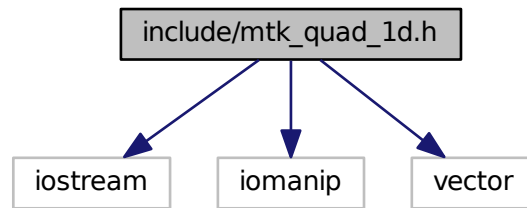
Includes the definition of the class Quad1D.

```

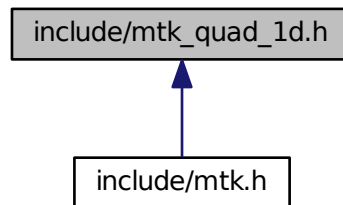
#include <iostream>
#include <iomanip>
#include <vector>

```

Include dependency graph for mtk\_quad\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::Quad1D](#)  
*Implements a 1D mimetic quadrature.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.55.1 Detailed Description

This class implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

See also

[mtk::Grad1D](#)

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Implement this class.

Definition in file [mtk\\_quad\\_1d.h](#).

## 18.56 mtk\_quad\_1d.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_QUAD_1D_H_
00062 #define MTK_INCLUDE_QUAD_1D_H_
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #include <vector>
00068
00069 namespace mtk {
00070
00081 class Quad1D {
00082 public:
00084 friend std::ostream& operator <<(std::ostream& stream, Quad1D &in);
00085
00087 Quad1D();
00088
00094 Quad1D(const Quad1D &quad);
00095

```

```

00097 ~Quad1D();
00098
00104 int degree_approximation() const;
00105
00111 Real *weights() const;
00112
00121 Real Integrate(Real (*Integrand)(Real xx), UniStgGrid1D grid) const;
00122
00123 private:
00124 int degree_approximation_;
00125
00126 std::vector<Real> weights_;
00127 };
00128 }
00129 #endif // End of: MTK_INCLUDE_QUAD_1D_H_

```

## 18.57 include/mtk\_robin\_bc\_descriptor\_1d.h File Reference

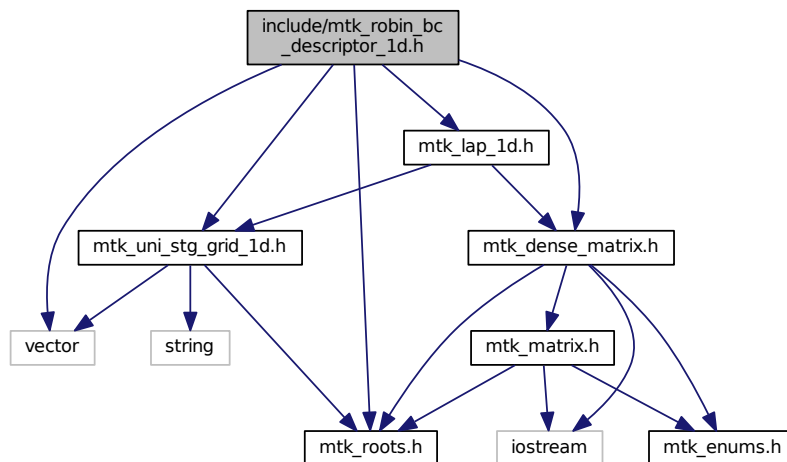
Impose Robin boundary conditions on the operators and on the grids.

```

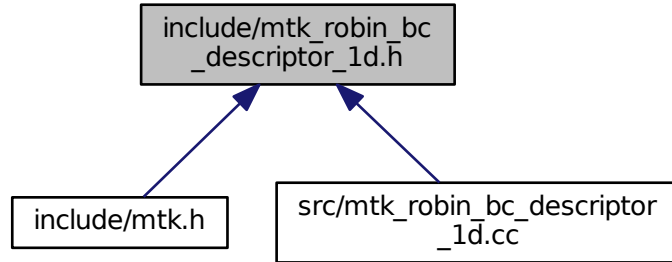
#include <vector>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_lap_1d.h"

```

Include dependency graph for mtk\_robin\_bc\_descriptor\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::RobinBCDescriptor1D`  
*Impose Robin boundary conditions on the operators and on the grids.*

## Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

## Typedefs

- typedef `Real(* mtk::CoefficientFunction0D)(const Real &tt)`  
*A function of a BC coefficient evaluated on a 0D domain and time.*

### 18.57.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 1D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context (  $\partial\Omega = \{a, b\} \subset \mathbb{R}$  ), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$



$$\delta_b(b,t)u(b,t) + \eta_b(b,t)u'(b,t) = \beta_b(b,t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_1d.h](#).

## 18.58 mtk\_robin\_bc\_descriptor\_1d.h

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <vector>
00090
00091 #include "mtk_roots.h"
00092 #include "mtk_dense_matrix.h"
00093 #include "mtk_uni_stg_grid_1d.h"

```

```

00094 #include "mtk_lap_1d.h"
00095
00096 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00097 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00098
00099 namespace mtk {
00111 typedef Real (*CoefficientFunction0D) (const Real &tt);
00112
00155 class RobinBCDescriptor1D {
00156 public:
00158 RobinBCDescriptor1D();
00159
00165 RobinBCDescriptor1D(const RobinBCDescriptor1D &desc);
00166
00168 ~RobinBCDescriptor1D() noexcept;
00169
00175 int highest_order_diff_west() const noexcept;
00176
00182 int highest_order_diff_east() const noexcept;
00183
00189 void PushBackWestCoeff(CoefficientFunction0D cw);
00190
00196 void PushBackEastCoeff(CoefficientFunction0D ce);
00197
00203 void set_west_condition(Real (*west_condition) (const
Real &tt)) noexcept;
00204
00210 void set_east_condition(Real (*east_condition) (const
Real &tt)) noexcept;
00211
00221 bool ImposeOnLaplacianMatrix(const Lap1D &lap,
00222 DenseMatrix &matrix,
00223 const Real &time = mtk::kZero) const;
00230 void ImposeOnGrid(UniStgGrid1D &grid, const Real &time =
mtk::kZero) const;
00231
00232 private:
00233 int highest_order_diff_west_;
00234 int highest_order_diff_east_;
00235
00236 std::vector<CoefficientFunction0D> west_coefficients_;
00237 std::vector<CoefficientFunction0D> east_coefficients_;
00238
00239 Real (*west_condition_) (const Real &tt);
00240 Real (*east_condition_) (const Real &tt);
00241 };
00242 }
00243 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_

```

## 18.59 include/mtk\_robin\_bc\_descriptor\_2d.h File Reference

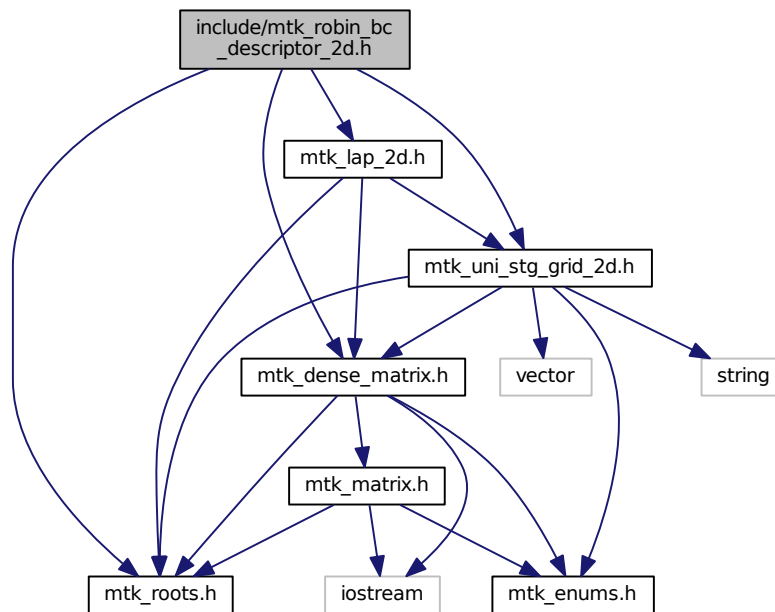
Impose Robin boundary conditions on the operators and on the grids.

```

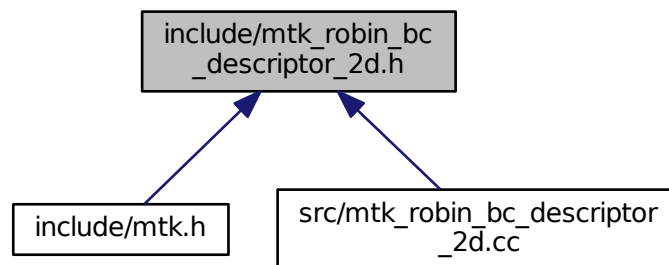
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk\_robin\_bc\_descriptor\_2d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::RobinBCDescriptor2D](#)

*Impose Robin boundary conditions on the operators and on the grids.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Typedefs

- typedef Real(\* [mtk::CoefficientFunction1D](#) )(const Real &xx, const Real &tt)

*A function of a BC coefficient evaluated on a 1D domain and time.*

### 18.59.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 2D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_2d.h](#).

### 18.60 mtk\_robin\_bc\_descriptor\_2d.h

```
00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
```

```

00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_
00081 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_
00082
00083 #include "mtk_roots.h"
00084 #include "mtk_dense_matrix.h"
00085 #include "mtk_lap_2d.h"
00086 #include "mtk_uni_stg_grid_2d.h"
00087
00088 namespace mtk{
00089
00097 typedef Real (*CoefficientFunction1D) (const Real &xx, const
 Real &tt);
00098
00132 class RobinBCDescriptor2D {
00133 public:
00135 RobinBCDescriptor2D();
00136
00142 RobinBCDescriptor2D(const RobinBCDescriptor2D &desc);
00143
00145 ~RobinBCDescriptor2D() noexcept;
00146
00152 int highest_order_diff_west() const noexcept;
00153
00159 int highest_order_diff_east() const noexcept;
00160
00166 int highest_order_diff_south() const noexcept;
00167
00173 int highest_order_diff_north() const noexcept;
00174
00181 void PushBackWestCoeff(CoefficientFunction1D cw);
00182
00189 void PushBackEastCoeff(CoefficientFunction1D ce);
00190
00197 void PushBackSouthCoeff(CoefficientFunction1D cs);
00198
00205 void PushBackNorthCoeff(CoefficientFunction1D cn);
00206
00213 void set_west_condition(Real (*west_condition) (const
 Real &yy,
00214 const Real &tt)) noexcept;
00215
00222 void set_east_condition(Real (*east_condition) (const
 Real &yy,
00223 const Real &tt)) noexcept;
00224
00231 void set_south_condition(Real (*south_condition) (const
 Real &xx,
00232 const Real &tt)) noexcept;
00233
00240 void set_north_condition(Real (*north_condition) (const
 Real &xx,

```

```

00241 const Real &tt)) noexcept;
00242
00251 bool ImposeOnLaplacianMatrix(const Lap2D &lap,
00252 const UniStgGrid2D &grid,
00253 DenseMatrix &matrix,
00254 const Real &time = kZero) const;
00261 void ImposeOnGrid(UniStgGrid2D &grid, const Real &time =
00262 kZero) const;
00263 private:
00272 bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00273 const UniStgGrid2D &grid,
00274 DenseMatrix &matrix,
00275 const Real &time = kZero) const;
00284 bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00285 const UniStgGrid2D &grid,
00286 DenseMatrix &matrix,
00287 const Real &time = kZero) const;
00296 bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00297 const UniStgGrid2D &grid,
00298 DenseMatrix &matrix,
00299 const Real &time = kZero) const;
00308 bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00309 const UniStgGrid2D &grid,
00310 DenseMatrix &matrix,
00311 const Real &time = kZero) const;
00320 bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00321 const UniStgGrid2D &grid,
00322 DenseMatrix &matrix,
00323 const Real &time = kZero) const;
00332 bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00333 const UniStgGrid2D &grid,
00334 DenseMatrix &matrix,
00335 const Real &time = kZero) const;
00344 bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00345 const UniStgGrid2D &grid,
00346 DenseMatrix &matrix,
00347 const Real &time = kZero) const;
00356 bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00357 const UniStgGrid2D &grid,
00358 DenseMatrix &matrix,
00359 const Real &time = kZero) const;
00360
00361 int highest_order_diff_west_;
00362 int highest_order_diff_east_;
00363 int highest_order_diff_south_;
00364 int highest_order_diff_north_;
00365
00366 std::vector<CoefficientFunction1D> west_coefficients_;
00367 std::vector<CoefficientFunction1D> east_coefficients_;
00368 std::vector<CoefficientFunction1D> south_coefficients_;
00369 std::vector<CoefficientFunction1D> north_coefficients_;
00370
00371 Real (*west_condition_)(const Real &xx, const Real &tt);
00372 Real (*east_condition_)(const Real &xx, const Real &tt);
00373 Real (*south_condition_)(const Real &yy, const Real &tt);
00374 Real (*north_condition_)(const Real &yy, const Real &tt);
00375 };
00376 }
00377 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_

```

## 18.61 include/mtk\_robin\_bc\_descriptor\_3d.h File Reference

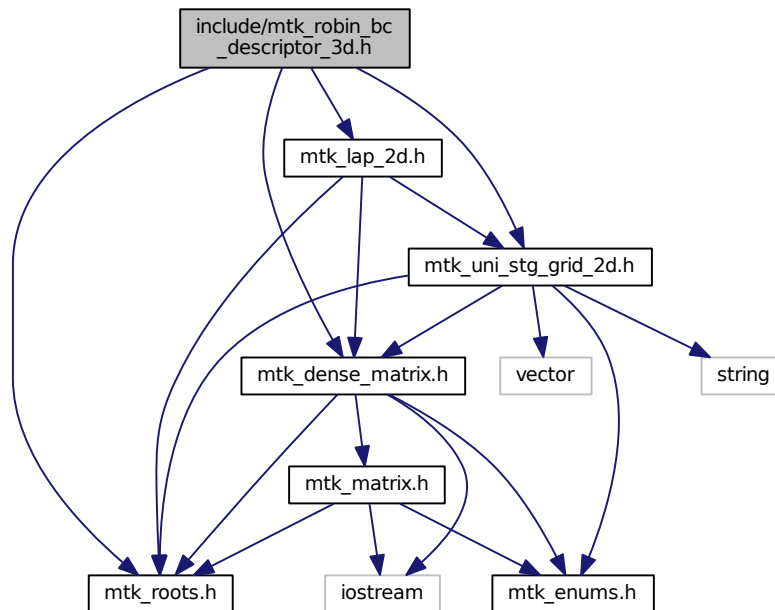
Impose Robin boundary conditions on the operators and on the grids.

```

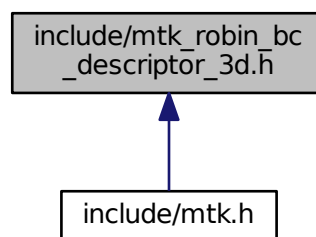
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk\_robin\_bc\_descriptor\_3d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::RobinBCDescriptor3D`

*Impose Robin boundary conditions on the operators and on the grids.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Typedefs

- typedef Real(\* [mtk::CoefficientFunction2D](#) )(const Real &xx, const Real &yy, const Real &tt)

*A function of a BC coefficient evaluated on a 2D domain and time.*

### 18.61.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_3d.h](#).

### 18.62 mtk\_robin\_bc\_descriptor\_3d.h

```
00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
```



```

00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00081 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00082
00083 #include "mtk_roots.h"
00084 #include "mtk_dense_matrix.h"
00085 #include "mtk_lap_2d.h"
00086 #include "mtk_uni_stg_grid_2d.h"
00087
00088 namespace mtk{
00089
00097 typedef Real (*CoefficientFunction2D) (const Real &xx,
00098 const Real &yy,
00099 const Real &tt);
00100
00134 class RobinBCDescriptor3D {
00135 public:
00137 RobinBCDescriptor3D();
00138
00144 RobinBCDescriptor3D(const RobinBCDescriptor3D &desc);
00145
00147 ~RobinBCDescriptor3D() noexcept;
00148
00154 int highest_order_diff_west() const noexcept;
00155
00156 // ...
00157
00164 void PushBackWestCoeff(CoefficientFunction2D cw);
00165
00166 // ...
00167
00174 void set_west_condition(Real (*west_condition) (const
00175 Real &xx,
00176 const Real &yy,
00177 const Real &tt)) noexcept;
00178 // ...
00179
00188 bool ImposeOnLaplacianMatrix(const Lap3D &lap,
00189 const UniStgGrid3D &grid,
00190 DenseMatrix &matrix,
00191 const Real &time = kZero) const;
00198 void ImposeOnGrid(UniStgGrid3D &grid, const Real &time =
00199 kZero) const;
00200 private:
00209 bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00210 const UniStgGrid2D &grid,
00211 DenseMatrix &matrix,
00212 const Real &time = kZero) const;
00221 bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00222 const UniStgGrid2D &grid,

```

```

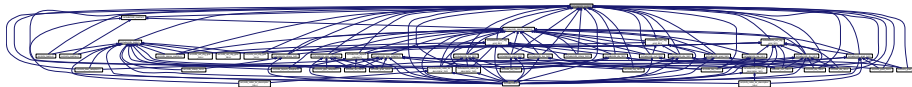
00223 DenseMatrix &matrix,
00224 const Real &time = kZero) const;
00233 bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00234 const UniStgGrid2D &grid,
00235 DenseMatrix &matrix,
00236 const Real &time = kZero) const;
00245 bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00246 const UniStgGrid2D &grid,
00247 DenseMatrix &matrix,
00248 const Real &time = kZero) const;
00257 bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00258 const UniStgGrid2D &grid,
00259 DenseMatrix &matrix,
00260 const Real &time = kZero) const;
00269 bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00270 const UniStgGrid2D &grid,
00271 DenseMatrix &matrix,
00272 const Real &time = kZero) const;
00281 bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00282 const UniStgGrid2D &grid,
00283 DenseMatrix &matrix,
00284 const Real &time = kZero) const;
00293 bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00294 const UniStgGrid2D &grid,
00295 DenseMatrix &matrix,
00296 const Real &time = kZero) const;
00297
00298 int highest_order_diff_west_;
00299 int highest_order_diff_east_;
00300 int highest_order_diff_south_;
00301 int highest_order_diff_north_;
00302 int highest_order_diff_bottom_;
00303 int highest_order_diff_top_;
00304
00305 std::vector<CoefficientFunction2D> west_coefficients_;
00306 std::vector<CoefficientFunction2D> east_coefficients_;
00307 std::vector<CoefficientFunction2D> south_coefficients_;
00308 std::vector<CoefficientFunction2D> north_coefficients_;
00309 std::vector<CoefficientFunction2D> bottom_coefficients_;
00310 std::vector<CoefficientFunction2D> top_coefficients_;
00311
00312 Real (*west_condition_)(const Real &xx,
00313 const Real &yy,
00314 const Real &tt);
00315 Real (*east_condition_)(const Real &xx,
00316 const Real &yy,
00317 const Real &tt);
00318 Real (*south_condition_)(const Real &xx,
00319 const Real &yy,
00320 const Real &tt);
00321 Real (*north_condition_)(const Real &xx,
00322 const Real &yy,
00323 const Real &tt);
00324 Real (*bottom_condition_)(const Real &xx,
00325 const Real &yy,
00326 const Real &tt);
00327 Real (*top_condition_)(const Real &xx,
00328 const Real &yy,
00329 const Real &tt);
00330 };
00331 }
00332 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_

```

## 18.63 include/mtk\_roots.h File Reference

Fundamental definitions to be used across all classes of the MTK.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Typedefs

- typedef float [mtk::Real](#)

*Users can simply change this to build a double- or single-precision MTK.*

## Variables

- const float [mtk::kZero](#) {0.0f}

*MTK's zero defined according to selective compilation.*

- const float [mtk::kOne](#) {1.0f}

*MTK's one defined according to selective compilation.*

- const float [mtk::kTwo](#) {2.0f}

*MTK's two defined according to selective compilation.*

- const float [mtk::kDefaultTolerance](#) {1e-7f}

*Considered tolerance for comparisons in numerical methods.*

- const float [mtk::kDefaultMimeticThreshold](#) {1e-6f}

*Default tolerance for higher-order mimetic operators.*

- const int [mtk::kDefaultOrderAccuracy](#) {2}

*Default order of accuracy for mimetic operators.*

- const int [mtk::kCriticalOrderAccuracyGrad](#) {10}

*At this order (and higher) we must use the CBSA to construct gradients.*

- const int [mtk::kCriticalOrderAccuracyDiv](#) {8}

*At this order (and higher) we must use the CBSA to construct divergences.*

### 18.63.1 Detailed Description

This file contains the fundamental definitions that classes of the MTK rely on to be implemented. Examples of these definitions are the definition of fundamental data types, and global variables affecting the construction of mimetic operators, among others.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at sciences dot sdsu dot edu

**Todo** Test selective precision mechanisms.

Definition in file [mtk\\_roots.h](#).

## 18.64 mtk\_roots.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_ROOTS_H_
00062 #define MTK_INCLUDE_ROOTS_H_
00063
00064 namespace mtk {
00070
00090 #ifdef MTK_PRECISION_DOUBLE
00091 typedef double Real;
00092 #else
00093 typedef float Real;
00094 #endif
00095
00121 #ifdef MTK_PRECISION_DOUBLE
00122 const double kZero{0.0};
00123 const double kOne{1.0};
00124 const double kTwo{2.0};
00125 #else
00126 const float kZero{0.0f};
00127 const float kOne{1.0f};
00128 const float kTwo{2.0f};
00129 #endif
00130
00140 #ifdef MTK_PRECISION_DOUBLE
00141 const double kDefaultTolerance{1e-7};
00142 #else
00143 const float kDefaultTolerance{1e-7f};
00144 #endif
00145
00155 #ifdef MTK_PRECISION_DOUBLE
00156 const double kDefaultMimeticThreshold{1e-6};
00157 #else
00158 const float kDefaultMimeticThreshold{1e-6f};

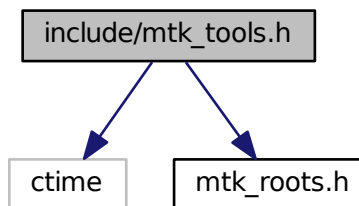
```

```
00159 #endif
00160
00168 const int kDefaultOrderAccuracy{2};
00169
00177 const int kCriticalOrderAccuracyGrad{10};
00178
00186 const int kCriticalOrderAccuracyDiv{8};
00187 }
00188 #endif // End of: MTK_INCLUDE_ROOTS_H_
```

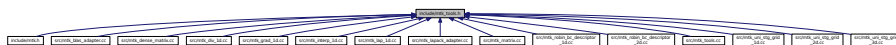
## 18.65 include/mtk\_tools.h File Reference

Tool manager class.

```
#include <ctime>
#include "mtk_roots.h"
Include dependency graph for mtk_tools.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::Tools`  
*Tool manager class.*

## Namespaces

- `mtk`  
*Mimetic Methods Toolkit namespace.*

### 18.65.1 Detailed Description

Definition of a class providing basic tools to ensure execution correctness, and to assists with unitary testing.

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Note**

Performance Tip 8.1. If they do not need to be modified by the called function, pass large objects using pointers to constant data or references to constant data, to obtain the performance benefits of pass-by-reference.

Definition in file [mtk\\_tools.h](#).

**18.66 mtk\_tools.h**

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_TOOLS_H_
00062 #define MTK_INCLUDE_TOOLS_H_
00063
00064 #include <ctime>
00065
00066 #include "mtk_roots.h"
00067
00068 namespace mtk {
00069
00080 class Tools {
00081 public:
00092 static void Prevent(const bool complement,
00093 const char *const fname,

```

```

00094 int lineno,
00095 const char *const fxname) noexcept;
00096
00102 static void BeginUnitTestNo(const int &nn) noexcept;
00103
00109 static void EndUnitTestNo(const int &nn) noexcept;
00110
00116 static void Assert(const bool &condition) noexcept;
00117
00118 private:
00119 static int test_number_;
00120
00121 static Real duration_;
00122
00123 static clock_t begin_time_;
00124 };
00125
00126 #endif // End of: MTK_INCLUDE_TOOLS_H_

```

## 18.67 include/mtk\_uni\_stg\_grid\_1d.h File Reference

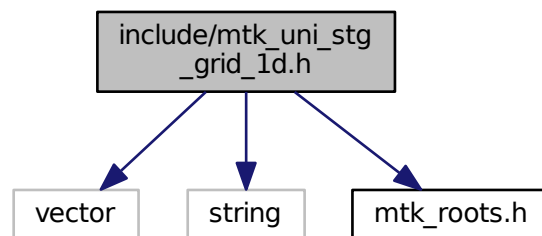
Definition of an 1D uniform staggered grid.

```

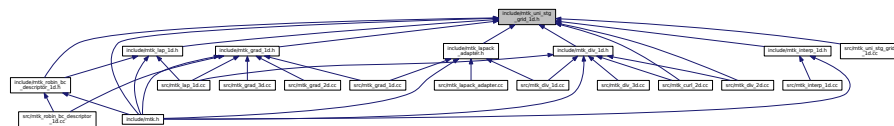
#include <vector>
#include <string>
#include "mtk_roots.h"

```

Include dependency graph for mtk\_uni\_stg\_grid\_1d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `mtk::UniStgGrid1D`  
*Uniform 1D Staggered Grid.*

## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

### 18.67.1 Detailed Description

Definition of an 1D uniform staggered grid.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Create overloaded binding routines that read data from files.

Definition in file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

### 18.68 mtk\_uni\_stg\_grid\_1d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_

```



```

00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078 public:
00080 friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
00083 UniStgGrid1D();
00084
00090 UniStgGrid1D(const UniStgGrid1D &grid);
00091
00102 UniStgGrid1D(const Real &west_bndy_x,
00103 const Real &east_bndy_x,
00104 const int &num_cells_x,
00105 const mtk::FieldNature &nature = mtk::SCALAR);
00106
00108 ~UniStgGrid1D();
00109
00115 Real west_bndy_x() const;
00116
00122 Real east_bndy_x() const;
00123
00129 Real delta_x() const;
00130
00138 const Real *discrete_domain_x() const;
00139
00147 Real *discrete_field();
00148
00154 int num_cells_x() const;
00155
00161 void BindScalarField(Real (*ScalarField)(const Real &xx));
00162
00173 void BindVectorField(Real (*VectorField)(Real xx));
00174
00186 bool WriteToFile(std::string filename,
00187 std::string space_name,
00188 std::string field_name) const;
00189
00190 private:
00191 FieldNature nature_;
00192
00193 std::vector<Real> discrete_domain_x_;
00194 std::vector<Real> discrete_field_;
00195
00196 Real west_bndy_x_;
00197 Real east_bndy_x_;
00198 Real num_cells_x_;
00199 Real delta_x_;
00200 };
00201
00202 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_

```

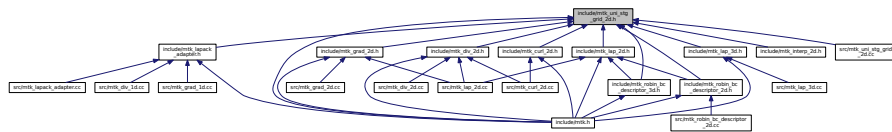
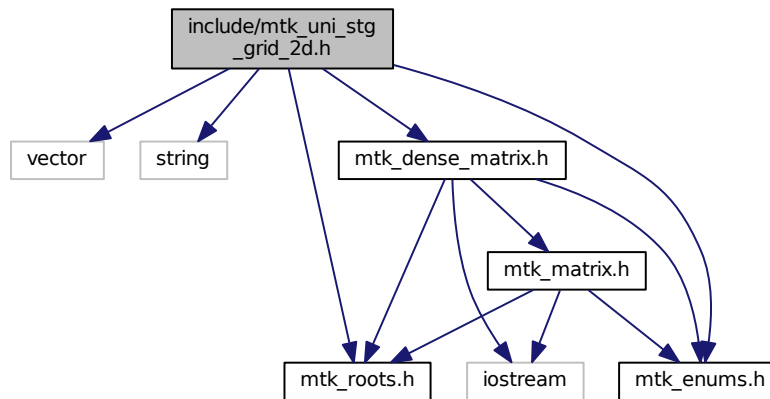
## 18.69 include/mtk\_uni\_stg\_grid\_2d.h File Reference

Definition of an 2D uniform staggered grid.

```

#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"

```



- class `mtk::UniStgGrid2D`  
*Uniform 2D Staggered Grid.*

- `mtk`  
*Mimetic Methods Toolkit namespace.*

### Definition of an 2D uniform staggered grid.

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

## 18.70 mtk\_uni\_stg\_grid\_2d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_2D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_2D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {
00069
00070 class UniStgGrid2D {
00080 public:
00082 friend std::ostream& operator <<(std::ostream& stream, UniStgGrid2D &in);
00083
00085 UniStgGrid2D();
00086
00092 UniStgGrid2D(const UniStgGrid2D &grid);
00093
00107 UniStgGrid2D(const Real &west_bndy_x,
00108 const Real &east_bndy_x,
00109 const int &num_cells_x,
00110 const Real &south_bndy_y,
00111 const Real &north_bndy_y,
00112 const int &num_cells_y,
00113 const mtk::FieldNature &nature =
00114 mtk::SCALAR);
00116 ~UniStgGrid2D();
00117

```

```

00125 const Real *discrete_domain_x() const;
00126
00134 const Real *discrete_domain_y() const;
00135
00141 Real *discrete_field();
00142
00150 FieldNature nature() const;
00151
00157 Real west_bndy() const;
00158
00164 Real east_bndy() const;
00165
00171 int num_cells_x() const;
00172
00178 Real delta_x() const;
00179
00185 Real south_bndy() const;
00186
00192 Real north_bndy() const;
00193
00199 int num_cells_y() const;
00200
00206 Real delta_y() const;
00207
00213 bool Bound() const;
00214
00220 int Size() const;
00221
00227 void BindScalarField(Real (*ScalarField)(const Real &xx, const
Real &yy));
00228
00242 void BindVectorField(Real (*VectorFieldPComponent)(const
Real &xx,
00243 const Real &yy),
00244 Real (*VectorFieldQComponent)(const Real &xx,
00245 const Real &yy));
00246
00259 bool WriteToFile(std::string filename,
00260 std::string space_name_x,
00261 std::string space_name_y,
00262 std::string field_name) const;
00263
00264 private:
00276 void BindVectorFieldPComponent(
00277 Real (*VectorFieldPComponent)(const Real &xx, const Real &yy));
00278
00290 void BindVectorFieldQComponent(
00291 Real (*VectorFieldQComponent)(const Real &xx, const Real &yy));
00292
00293 std::vector<Real> discrete_domain_x;
00294 std::vector<Real> discrete_domain_y;
00295 std::vector<Real> discrete_field;
00296
00297 FieldNature nature_;
00298
00299 Real west_bndy_;
00300 Real east_bndy_;
00301 int num_cells_x_;
00302 Real delta_x_;
00303
00304 Real south_bndy_;
00305 Real north_bndy_;
00306 int num_cells_y_;
00307 Real delta_y_;
00308 };
00309 }
00310 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_

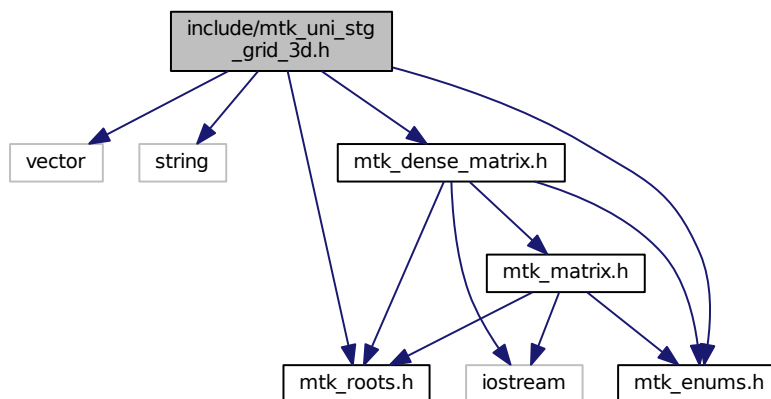
```

## 18.71 include/mtk\_uni\_stg\_grid\_3d.h File Reference

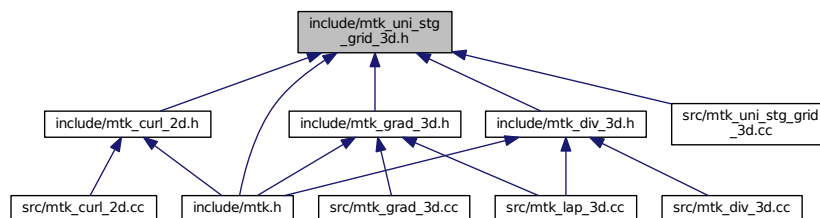
Definition of an 3D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk\_uni\_stg\_grid\_3d.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [mtk::UniStgGrid3D](#)  
*Uniform 3D Staggered Grid.*

## Namespaces

- [mtk](#)  
*Mimetic Methods Toolkit namespace.*

### 18.71.1 Detailed Description

Definition of an 3D uniform staggered grid.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Create overloaded binding routines that read data from files.

Definition in file [mtk\\_uni\\_stg\\_grid\\_3d.h](#).

## 18.72 mtk\_uni\_stg\_grid\_3d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_3D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_3D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {

```

```

00069
00079 class UniStgGrid3D {
00080 public:
00082 friend std::ostream& operator <<(std::ostream& stream, UniStgGrid3D &in);
00083
00091 UniStgGrid3D operator=(const UniStgGrid3D &in);
00092
00094 UniStgGrid3D();
00095
00101 UniStgGrid3D(const UniStgGrid3D &grid);
00102
00119 UniStgGrid3D(const Real &west_bndy_x,
00120 const Real &east_bndy_x,
00121 const int &num_cells_x,
00122 const Real &south_bndy_y,
00123 const Real &north_bndy_y,
00124 const int &num_cells_y,
00125 const Real &bottom_bndy_z,
00126 const Real &top_bndy_z,
00127 const int &num_cells_z,
00128 const mtk::FieldNature &nature =
mtk::SCALAR);
00129
00131 ~UniStgGrid3D();
00132
00140 const Real *discrete_domain_x() const;
00141
00149 const Real *discrete_domain_y() const;
00150
00158 const Real *discrete_domain_z() const;
00159
00165 Real *discrete_field();
00166
00174 FieldNature nature() const;
00175
00181 Real west_bndy() const;
00182
00188 Real east_bndy() const;
00189
00195 int num_cells_x() const;
00196
00202 Real delta_x() const;
00203
00209 Real south_bndy() const;
00210
00216 Real north_bndy() const;
00217
00223 int num_cells_y() const;
00224
00230 Real delta_y() const;
00231
00237 Real bottom_bndy() const;
00238
00244 Real top_bndy() const;
00245
00251 int num_cells_z() const;
00252
00258 Real delta_z() const;
00259
00265 bool Bound() const;
00266
00272 int Size() const;
00273
00279 void BindScalarField(
00280 Real (*ScalarField)(const Real &xx, const Real &yy, const Real &zz));
00281
00298 void BindVectorField(Real (*VectorFieldPComponent)(const
Real &xx,
00299 const Real &yy,
00300 const Real &zz),
00301 Real (*VectorFieldQComponent)(const Real &xx,
00302 const Real &yy,
00303 const Real &zz),
00304 Real (*VectorFieldRComponent)(const Real &xx,
00305 const Real &yy,
00306 const Real &zz));
00307
00321 bool WriteToFile(std::string filename,
00322 std::string space_name_x,
00323 std::string space_name_y,
00324 std::string space_name_z,

```

```

00325 std::string field_name) const;
00326
00327 private:
00340 void BindVectorFieldPComponent (
00341 Real (*VectorFieldPComponent) (const Real &xx,
00342 const Real &yy,
00343 const Real &zz));
00344
00357 void BindVectorFieldQComponent (
00358 Real (*VectorFieldQComponent) (const Real &xx,
00359 const Real &yy,
00360 const Real &zz));
00361
00374 void BindVectorFieldRComponent (
00375 Real (*VectorFieldRComponent) (const Real &xx,
00376 const Real &yy,
00377 const Real &zz));
00378
00379 std::vector<Real> discrete_domain_x_;
00380 std::vector<Real> discrete_domain_y_;
00381 std::vector<Real> discrete_domain_z_;
00382 std::vector<Real> discrete_field_;
00383
00384 FieldNature nature_;
00385
00386 Real west_bndy_;
00387 Real east_bndy_;
00388 int num_cells_x_;
00389 Real delta_x_;
00390
00391 Real south_bndy_;
00392 Real north_bndy_;
00393 int num_cells_y_;
00394 Real delta_y_;
00395
00396 Real bottom_bndy_;
00397 Real top_bndy_;
00398 int num_cells_z_;
00399 Real delta_z_;
00400 };
00401 }
00402 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_3D_H_

```

## 18.73 Makefile.inc File Reference

### 18.74 Makefile.inc

```

00001 # Makefile setup file for the MTK.
00002
00003 SHELL := /bin/bash
00004
00005 # 1. Absolute path to base directory of the MTK.
00006 # _____
00007
00008 BASE = /home/esanchez/Dropbox/MTK
00009
00010 # 2. The machine (platform) identifier and required machine precision.
00011 # _____
00012
00013 # Options are:
00014 # - LINUX: A LINUX box installation.
00015 # - OSX: Uses OS X optimized solvers.
00016
00017 PLAT = LINUX
00018
00019 # Options are:
00020 # - SINGLE: Use 4 B floating point numbers.
00021 # - DOUBLE: Use 8 B floating point numbers.
00022
00023 PRECISION = DOUBLE
00024
00025 # 3. Optimized solvers and operations by means of ATLAS in Linux?
00026 # _____
00027
00028 # If you have selected OSX in step 1, then you don't need to worry about this.

```



```

00029
00030 # Options are ON xor OFF:
00031
00032 ATL_OPT = OFF
00033
00034 # 4. Paths to dependencies (header files for compiling).
00035 # _____
00036
00037 # GLPK include path (soon to go):
00038
00039 GLPK_INC = $(HOME)/Libraries/glpk-4.35/include
00040
00041 # Linux: If ATLAS optimization is ON, users should only provide the path to
00042 # ATLAS:
00043
00044 ATLAS_INC = $(HOME)/Libraries/ATLAS_3.8.4-CORE/include
00045
00046 # OS X: Do nothing.
00047
00048 # 5. Paths to dependencies (archive files for (static) linking).
00049 # _____
00050
00051 # GLPK linking path (soon to go):
00052
00053 GLPK_LIB = $(HOME)/Libraries/glpk-4.35/lib/lib64/libglpk.a
00054
00055 # If optimization is OFF, then provide the paths for:
00056
00057 BLAS_LIB = $(HOME)/Libraries/BLAS-3.5.0/libblas.a
00058 LAPACK_LIB = $(HOME)/Libraries/lapack-3.5.0/liblapack.a
00059
00060 # WARNING: Vendor libraries should be used whenever they are available.
00061
00062 # However, if optimization is ON, please provide the path the ATLAS' archive:
00063
00064 ATLAS_LIB = $(HOME)/Libraries/ATLAS_3.8.4-CORE/ATLAS_3.8.4-BUILD-Citadel/lib
00065
00066 # 6. Compiler and its flags.
00067 # _____
00068
00069 CC = g++
00070
00071 # Selective Verbose Execution for Quick Debugging. Options are defined per
00072 # concern, and per data hierarchy on each concern.
00073
00074 # 0: NO verbose at all.
00075
00076 # 1: Enable verbose down to the 7th concern: messages.
00077 # 2: Enable verbose down to the 7th concern: messages + scalar results.
00078 # 3: Enable verbose down to the 7th concern. 1.1. + array results.
00079 # 4: Enable verbose down to the 7th concern. 1.2. + matrix results.
00080
00081 # 5: Enable verbose down to the 6th concern: messages.
00082 # 6: Enable verbose down to the 6th concern: messages + scalar results.
00083 # 7: Enable verbose down to the 6th concern. 2.1. + array results.
00084 # 8: Enable verbose down to the 6th concern. 2.2. + matrix results.
00085
00086 # 9: Enable verbose down to the 5th concern: messages.
00087 # 10: Enable verbose down to the 5th concern: messages + scalar results.
00088 # 11: Enable verbose down to the 5th concern. 3.1. + array results.
00089 # 12: Enable verbose down to the 5th concern. 3.2. + matrix results.
00090
00091 # 13: Enable verbose down to the 4th concern: messages.
00092 # 14: Enable verbose down to the 4th concern: messages + scalar results.
00093 # 15: Enable verbose down to the 4th concern. 4.1. + array results.
00094 # 16: Enable verbose down to the 4th concern. 4.2. + matrix results.
00095
00096 VERBOSE_LEVEL = 16
00097
00098 # Enable preventions. In the MTK, methods first validate their required
00099 # pre-conditions in run-time. Similarly, in many points throughout the MTK
00100 # codebase, different sanity checks are performed, as well. If this symbol is
00101 # defined to be 0, the MTK will # perform no validations to enhance execution
00102 # performance. Options are:
00103 # - YES.
00104 # - NO.
00105
00106 PERFORM_PREVENTIONS = YES
00107
00108 # Enables creation of LaTeX tables verbosing the computation of mimetic weights.
00109

```

```

00110 VERBOSE_WEIGHTS = YES
00111
00112 # Flags recommended for release code:
00113
00114 CCFLAGS = -Wall -Werror -O2
00115
00116 # Flags recommended for debugging code:
00117
00118 CCFLAGS = -Wall -Werror -g
00119
00120 # 7. Archiver, its flags, and ranlib:
00121 #
00122
00123 ARCH = ar
00124 ARCHFLAGS = cr
00125
00126 # If your system does not have "ranlib" then set: "RANLIB = echo":
00127
00128 RANLIB = echo
00129
00130 # But, if possible:
00131
00132 RANLIB = ranlib
00133
00134 # 8. Valgrind's memcheck options (optional):
00135 #
00136
00137 MEMCHECK_OPTS = -v --tool=memcheck --leak-check=full --show-leak-kinds=all \
00138 --track-origins=yes --freelist-vol=20000000
00139
00140 # Done! User, please, do not mess with the definitions from this point on.
00141
00142 #
00143 #
00144 #
00145
00146 # MTK-related.
00147 #
00148
00149 SRC = $(BASE)/src
00150 INCLUDE = $(BASE)/include
00151 LIB = $(BASE)/lib
00152 MTK_LIB = $(LIB)/libmtk.a
00153 TESTS = $(BASE)/tests
00154 EXAMPLES = $(BASE)/examples
00155
00156 # Compiling-related.
00157 #
00158
00159 CCFLAGS += -std=c++11 -fPIC \
00160 -DMTK_VERBOSE_LEVEL=$(VERBOSE_LEVEL) -I$(INCLUDE) -c
00161
00162 ifeq ($(PRECISION),DOUBLE)
00163 CCFLAGS += -DMTK_PRECISION_DOUBLE
00164 else
00165 CCFLAGS += -DMTK_PRECISION_SINGLE
00166 endif
00167
00168 ifeq ($(PERFORM_PREVENTIONS),YES)
00169 CCFLAGS += -DMTK_PERFORM_PREVENTIONS
00170 endif
00171
00172 ifeq ($(VERBOSE_WEIGHTS),YES)
00173 CCFLAGS += -DMTK_VERBOSE_WEIGHTS
00174 endif
00175
00176 # Only the GLPK is included because the other dependencies are coded in Fortran.
00177
00178 ifeq ($(ATL_OPT),ON)
00179 CCFLAGS += -I$(GLPK_INC) $(ATLAS_INC)
00180 else
00181 CCFLAGS += -I$(GLPK_INC)
00182 endif
00183
00184 # Linking-related.
00185 #
00186
00187 NOOPT_LIBS = $(LAPACK_LIB) $(BLAS_LIB) -lm $(GLPK_LIB) -lstdc++
00188
00189 OPT_LIBS = -L$(ATLAS_LIB) -latlas -llapack -lblas -lm -latlas -lstdc++
00190

```

```

00191 ifeq ($(PLAT),OSX)
00192 LINKER = g++
00193 LINKER += -framework Accelerate $(GLPK_LIB) $(MTK_LIB)
00194 else
00195 ifeq ($(ATL_OPT),ON)
00196 LINKER = g++
00197 LIBS = $(MTK_LIB)
00198 LIBS += $(OPT_LIBS)
00199 else
00200 LINKER = gfortran
00201 LIBS = $(MTK_LIB)
00202 LIBS += $(NOOPT_LIBS)
00203 endif
00204 endif
00205
00206 # Documentation-related.
00207 # -----
00208
00209 DOCGEN = doxygen
00210 DOCFEILNAME = doc_config.dxcf
00211 DOC = $(BASE)/doc
00212 DOCFEIL = $(BASE)/$(DOCFEILNAME)

```

## 18.75 README.md File Reference

### 18.76 README.md

```

00001 # The Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu**
00004
00005 ## 1. Description
00006
00007 We define numerical methods that are based on discretizations preserving the
00008 properties of their continuous counterparts to be mimetic.
00009
00010 The Mimetic Methods Toolkit (MTK) is a C++11 library for mimetic numerical
00011 methods. It is a set of classes for mimetic interpolation, mimetic
00012 quadratures, and mimetic finite difference methods for the numerical
00013 solution of ordinary and partial differential equations.
00014
00015 ## 2. Dependencies
00016
00017 This README file assumes all of these dependencies are installed in the
00018 following folder:
00019
00020 ```
00021 $(HOME)/Libraries/
00022 ```
00023
00024 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00025 routines for the internal computation on some of the layers. However, ATLAS
00026 requires both BLAS and LAPACK in order to create their optimized distributions.
00027 Therefore, the following dependencies tree arises:
00028
00029 ### For Linux:
00030
00031 1. LAPACK - Available from: http://www.netlib.org/lapack/
00032 1. BLAS - Available from: http://www.netlib.org/blas/
00033
00034 2. GLPK - Available from: https://www.gnu.org/software/glpk/
00035
00036 3. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00037 1. LAPACK - Available from: http://www.netlib.org/lapack/
00038 1. BLAS - Available from: http://www.netlib.org/blas/
00039
00040 4. (Optional) Valgrind - Available from: http://valgrind.org/
00041
00042 5. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00043
00044 ### For OS X:
00045
00046 1. GLPK - Available from: https://www.gnu.org/software/glpk/
00047
00048 ## 3. Installation

```

```

00049
00050 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00051
00052 The following steps are required to build and test the MTK. Please use the
00053 accompanying 'Makefile.inc' file, which should provide a solid template to
00054 start with. The following command provides help on the options for make:
00055
00056 ```
00057 $ make help
00058 -----
00059 Makefile for the MTK.
00060
00061 Options are:
00062 - all: builds the library, the tests, and examples.
00063 - mtklib: builds the library.
00064 - test: builds the test files.
00065 - example: builds the examples.
00066
00067 - testall: runs all the tests.
00068
00069 - gendoc: generates the documentation for the library.
00070
00071 - clean: cleans all the generated files.
00072 - cleanlib: cleans the generated archive and object files.
00073 - cleantest: cleans the generated tests executables.
00074 - cleanexample: cleans the generated examples executables.
00075 -----
00076 ```
00077
00078 ### PART 2. BUILD THE LIBRARY.
00079
00080 ```
00081 $ make
00082 ```
00083
00084 If successful you'll read (before building the tests and examples):
00085 ```
00086 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00087 ```
00088
00089 ## 4. Contact, Support, and Credits
00090
00091 The GitHub repository is: https://github.com/ejspeiro/MTK
00092
00093 The MTK is developed by researchers and adjuncts to the
00094 [Computational Science Research Center (CSRC)] (http://www.csrc.sdsu.edu/)
00095 at [San Diego State University (SDSU)] (http://www.sdsu.edu/).
00096
00097 Currently the developers are:
00098
00099 - **Eduardo J. Sanchez, PhD - esanchez@mail.sdsu.edu - @ejspeiro
00100 - Jose E. Castillo, PhD - jcastillo@mail.sdsu.edu
00101 - Guillermo F. Miranda, PhD - unigrav@hotmail.com
00102 - Christopher P. Paolini, PhD - paolini@engineering.sdsu.edu
00103 - Angel Boada.
00104 - Johnny Corbino.
00105 - Raul Vargas-Navarro.
00106
00107 ### 4.1. Acknowledgements and Contributions
00108
00109 The authors would like to acknowledge valuable advising, feedback,
00110 and actual contributions from research personnel at the Computational Science
00111 Research Center (CSRC) at San Diego State University (SDSU). Their input was
00112 important to the fruition of this work. Specifically, our thanks go to
00113 (alphabetical order):
00114
00115 -# Mohammad Abouali, PhD
00116 -# Dany De Cecchis, PhD
00117 -# Otilio Rojas, PhD
00118 -# Julia Rossi.
00119
00120 ## 5. Referencing This Work
00121
00122 Please reference this work as follows:
00123
00124 Please reference this work as follows:
00125 ```
00126 @article{Sanchez2014308,
00127 title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
00128 Finite Differences ",
00129 journal = "Journal of Computational and Applied Mathematics ",

```

```

00130 volume = "270",
00131 number = "",
00132 pages = "308 - 322",
00133 year = "2014",
00134 note = "Fourth International Conference on Finite Element Methods in
00135 Engineering and Sciences (FEMTEC 2013) ",
00136 issn = "0377-0427",
00137 doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
00138 url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
00139 author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
00140 keywords = "Object-oriented development",
00141 keywords = "Partial differential equations",
00142 keywords = "Application programming interfaces",
00143 keywords = "Mimetic Finite Differences "
00144 }
00145
00146 @Inbook{Sanchez2015,
00147 author="Sanchez, Eduardo and Paolini, Christopher and Blomgren, Peter
00148 and Castillo, Jose",
00149 editor="Kirby, M. Robert and Berzins, Martin and Hesthaven, S. Jan",
00150 chapter="Algorithms for Higher-Order Mimetic Operators",
00151 title="Spectral and High Order Methods for Partial Differential Equations
00152 ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014,
00153 Salt Lake City, Utah, USA",
00154 year="2015",
00155 publisher="Springer International Publishing",
00156 address="Cham",
00157 pages="425--434",
00158 isbn="978-3-319-19800-2",
00159 doi="10.1007/978-3-319-19800-2_39",
00160 url="http://dx.doi.org/10.1007/978-3-319-19800-2_39"
00161 }
00162 ```
00163
00164 Finally, please feel free to contact me with suggestions or corrections:
00165
00166 **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - @ejspeiro
00167
00168 Thanks and happy coding!

```

## 18.77 src/mtk\_blas\_adapter.cc File Reference

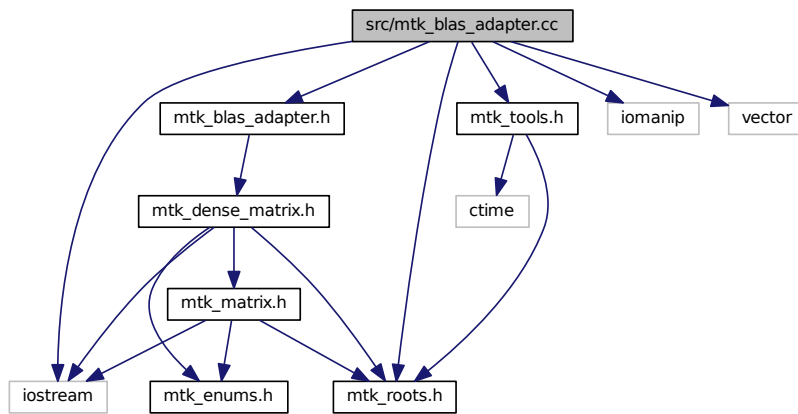
Adapter class for the BLAS API.

```

#include <iostream>
#include <iomanip>
#include <vector>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"

```

Include dependency graph for `mtk_blas_adapter.cc`:



## Namespaces

- `mtk`

*Mimetic Methods Toolkit namespace.*

## Functions

- float `mtk::snrm2_` (int \*n, float \*x, int \*incx)
- void `mtk::saxpy_` (int \*n, float \*sa, float \*sx, int \*incx, float \*sy, int \*incy)
- void `mtk::sgemv_` (char \*trans, int \*m, int \*n, float \*alpha, float \*a, int \*lda, float \*x, int \*incx, float \*beta, float \*y, int \*incy)
- void `mtk::sgemm_` (char \*transa, char \*transb, int \*m, int \*n, int \*k, double \*alpha, double \*a, int \*lda, double \*b, aamm int \*ldb, double \*beta, double \*c, int \*ldc)

### 18.77.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the BLAS.

The **BLAS (Basic Linear Algebra Subprograms)** are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.

The BLAS can be installed from links given in the See Also section of this page.

See also

<http://www.netlib.org/blas/>  
<https://software.intel.com/en-us/non-commercial-software-development>

**Todo** Write documentation using LaTeX.

## Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_blas\\_adapter.cc](#).

## 18.78 mtk\_blas\_adapter.cc

```

00001
00027 /*
00028 Copyright (C) 2015, Computational Science Research Center, San Diego State
00029 University. All rights reserved.
00030
00031 Redistribution and use in source and binary forms, with or without modification,
00032 are permitted provided that the following conditions are met:
00033
00034 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00035 and a copy of the modified files should be reported once modifications are
00036 completed, unless these modifications are made through the project's GitHub
00037 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00038 should be developed and included in any deliverable.
00039
00040 2. Redistributions of source code must be done through direct
00041 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00042
00043 3. Redistributions in binary form must reproduce the above copyright notice,
00044 this list of conditions and the following disclaimer in the documentation and/or
00045 other materials provided with the distribution.
00046
00047 4. Usage of the binary form on proprietary applications shall require explicit
00048 prior written permission from the the copyright holders, and due credit should
00049 be given to the copyright holders.
00050
00051 5. Neither the name of the copyright holder nor the names of its contributors
00052 may be used to endorse or promote products derived from this software without
00053 specific prior written permission.
00054
00055 The copyright holders provide no reassurances that the source code provided does
00056 not infringe any patent, copyright, or any other intellectual property rights of
00057 third parties. The copyright holders disclaim any liability to any recipient for
00058 claims brought against recipient by any third party for infringement of that
00059 parties intellectual property rights.
00060
00061 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00062 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00063 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00064 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00065 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00066 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00067 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00068 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00069 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00070 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00071 */
00072
00073 #include <iostream>
00074 #include <iomanip>
00075
00076 #include <vector>
00077
00078 #include "mtk_roots.h"
00079 #include "mtk_tools.h"
00080 #include "mtk_blas_adapter.h"
00081
00082 namespace mtk {
00083
00084 extern "C" {
00085
00086 #ifdef MTK_PRECISION_DOUBLE
00087
00100 double dnm2_(int *n, double *x, int *incx);
00101 #else
00102
00115 float snrm2_(int *n, float *x, int *incx);
00116 #endif
00117

```

```

00118 #ifdef MTK_PRECISION_DOUBLE
00119
00138 void daxpy_(int *n, double *da, double *dx, int *incx, double *dy, int *incy);
00139 #else
00140
00159 void saxpy_(int *n, float *sa, float *sx, int *incx, float *sy, int *incy);
00160 #endif
00161
00162 #ifdef MTK_PRECISION_DOUBLE
00163
00191 void dgemv_(char *trans,
00192 int *m,
00193 int *n,
00194 double *alpha,
00195 double *a,
00196 int *lda,
00197 double *x,
00198 int *incx,
00199 double *beta,
00200 double *y,
00201 int *incy);
00202 #else
00203
00231 void sgemv_(char *trans,
00232 int *m,
00233 int *n,
00234 float *alpha,
00235 float *a,
00236 int *lda,
00237 float *x,
00238 int *incx,
00239 float *beta,
00240 float *y,
00241 int *incy);
00242 #endif
00243
00244 #ifdef MTK_PRECISION_DOUBLE
00245
00270 void dgemm_(char *transa,
00271 char* transb,
00272 int *m,
00273 int *n,
00274 int *k,
00275 double *alpha,
00276 double *a,
00277 int *lda,
00278 double *b,
00279 int *ldb,
00280 double *beta,
00281 double *c,
00282 int *ldc);
00283 }
00284 #else
00285
00310 void sgemm_(char *transa,
00311 char* transb,
00312 int *m,
00313 int *n,
00314 int *k,
00315 double *alpha,
00316 double *a,
00317 int *lda,
00318 double *b, aamm
00319 int *ldb,
00320 double *beta,
00321 double *c,
00322 int *ldc);
00323 }
00324 #endif
00325 }
00326
00327 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00328
00329 #ifdef MTK_PERFORM_PREVENTIONS
00330 mtk::Tools::Prevent(in_length <= 0, __FILE__, __LINE__, __func__);
00331 #endif
00332
00333 int incx{1}; // Increment for the elements of xx. ix >= 0.
00334
00335 #ifdef MTK_PRECISION_DOUBLE
00336 return dnrn2_(&in_length, in, &incx);

```



```

00337 #else
00338 return snrm2_(&in_length, in, &incx);
00339 #endif
00340 }
00341
00342 void mtk::BLASAdapter::RealAXPY(mtk::Real alpha,
00343 mtk::Real *xx,
00344 mtk::Real *yy,
00345 int &in_length) {
00346
00347 #ifdef MTK_PERFORM_PREVENTIONS
00348 mtk::Tools::Prevent(xx == nullptr, __FILE__, __LINE__, __func__);
00349 mtk::Tools::Prevent(yy == nullptr, __FILE__, __LINE__, __func__);
00350 #endif
00351
00352 int incx{1}; // Increment for the elements of xx. ix >= 0.
00353
00354 #ifdef MTK_PRECISION_DOUBLE
00355 daxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00356 #else
00357 saxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00358 #endif
00359 }
00360
00361 mtk::Real mtk::BLASAdapter::RelNorm2Error(
00362 mtk::Real *computed,
00363 mtk::Real *known,
00364 int length) {
00365
00366 #ifdef MTK_PERFORM_PREVENTIONS
00367 mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00368 mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00369 #endif
00370
00371 mtk::Real norm_2_computed{mtk::BLASAdapter::RealNRM2(known, length)};
00372
00373 mtk::Real alpha{-mtk::kOne};
00374
00375 mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00376
00377 mtk::Real norm_2_difference{mtk::BLASAdapter::RealNRM2(computed,
00378 length)};
00379
00380 return norm_2_difference/norm_2_computed;
00381 }
00382
00383 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00384 mtk::DenseMatrix &aa,
00385 mtk::Real *xx,
00386 mtk::Real &beta,
00387 mtk::Real *yy) {
00388
00389 // Make sure input matrices are row-major ordered.
00390
00391 if (aa.matrix_properties().ordering() ==
00392 mtk::MatrixOrdering::COL_MAJOR) {
00393 aa.OrderRowMajor();
00394 }
00395
00396 char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00397
00398 int mm{aa.num_rows()}; // Rows of aa.
00399 int nn{aa.num_cols()}; // Columns of aa.
00400 int lda{(aa.matrix_properties()).ld()}; // Leading dimension.
00401 int incx{1}; // Increment of values in x.
00402 int incy{1}; // Increment of values in y.
00403
00404 std::swap(mm, nn);
00405 #ifdef MTK_PRECISION_DOUBLE
00406 dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00407 xx, &incx, &beta, yy, &incy);
00408 #else
00409 sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00410 xx, &incx, &beta, yy, &incy);
00411 #endif
00412 std::swap(mm, nn);
00413 }
00414
00415 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
00416 mtk::DenseMatrix &aa,
00417 mtk::DenseMatrix &bb) {

```

```

00414
00415 #ifdef MTK_PERFORM_PREVENTIONS
00416 mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00417 __FILE__, __LINE__, __func__);
00418 #endif
00419
00421 if (aa.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00422 aa.OrderRowMajor();
00423 }
00424 if (bb.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00425 bb.OrderRowMajor();
00426 }
00427
00429 char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00430 char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00431
00432 int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00433 int nn{bb.num_cols()}; // Cols of bb and cols of cc.
00434 int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00435
00436 int cc_num_rows{mm}; // Rows of cc.
00437 int cc_num_cols{nn}; // Columns of cc.
00438
00439 int lda{std::max(1, kk)}; // Leading dimension of the aa matrix.
00440 int ldb{std::max(1, nn)}; // Leading dimension of the bb matrix.
00441 int ldc{std::max(1, mm)}; // Leading dimension of the cc matrix.
00442
00443 mtk::Real alpha{mtk::kOne}; // First scalar coefficient.
00444 mtk::Real beta{mtk::kZero}; // Second scalar coefficient.
00445
00446 mtk::DenseMatrix cc_col_maj_ord(cc_num_rows, cc_num_cols); // Output matrix.
00447
00448 cc_col_maj_ord.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00449
00451 #ifdef MTK_PRECISION_DOUBLE
00452 dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00453 bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00454 #else
00455 sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00456 bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00457 #endif
00458
00459 #if MTK_VERBOSE_LEVEL > 12
00460 std::cout << "cc_col_maj_ord =" << std::endl;
00461 std::cout << cc_col_maj_ord << std::endl;
00462 #endif
00463
00464 cc_col_maj_ord.OrderRowMajor();
00465
00466 return cc_col_maj_ord;
00467 }
00468
00469 mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM(
mtk::Real alpha,
00470
00471 mtk::DenseMatrix &aa) {
00472
00473 #ifdef MTK_PERFORM_PREVENTIONS
00474 mtk::Tools::Prevent(aa.num_rows() == 0, __FILE__, __LINE__, __func__);
00475 mtk::Tools::Prevent(aa.num_cols() == 0, __FILE__, __LINE__, __func__);
00476 #endif
00477
00478 if (aa.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00479 aa.OrderRowMajor();
00480 }
00481
00483 char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00484 char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00485
00486 int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00487 int nn{aa.num_cols()}; // Cols of bb and cols of cc.
00488 int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00489
00490 int lda{std::max(1, kk)}; // Leading dimension of the aa matrix.
00491 int ldb{std::max(1, nn)}; // Leading dimension of the bb matrix.
00492 int ldc{std::max(1, mm)}; // Leading dimension of the cc matrix.
00493
00494 mtk::Real beta{alpha}; // Second scalar coefficient.
00495

```

```

00496 alpha = mtk::kZero;
00497
00498 mtk::DenseMatrix alpha_aa(aa); // Output matrix.
00499
00501 #ifdef MTK_PRECISION_DOUBLE
00502 dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lida,
00503 aa.data(), &lddb, &beta, alpha_aa.data(), &lcdc);
00504 #else
00505 sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lida,
00506 aa.data(), &lddb, &beta, alpha_aa.data(), &lcdc);
00507 #endif
00508
00509 #if MTK_VERBOSE_LEVEL > 12
00510 std::cout << "alpha_aa =" << std::endl;
00511 std::cout << alpha_aa << std::endl;
00512 #endif
00513
00514 return alpha_aa;
00515 }

```

## 18.79 src/mtk\_curl\_2d.cc File Reference

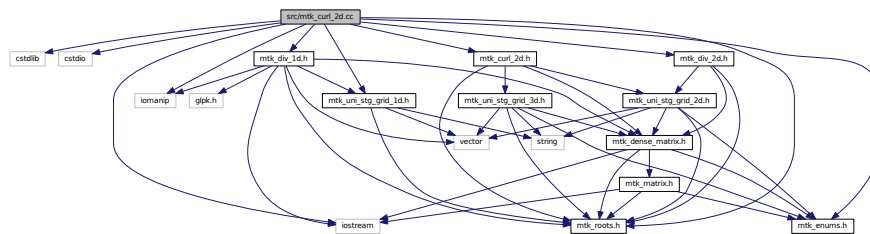
Implements the class Curl2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"

```

Include dependency graph for mtk\_curl\_2d.cc:



### 18.79.1 Detailed Description

This class implements a 2D curl matrix operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_curl\\_2d.cc](#).

## 18.80 mtk\_curl\_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_enums.h"
00065 #include "mtk_uni_stg_grid_ld.h"
00066 #include "mtk_div_ld.h"
00067 #include "mtk_div_2d.h"
00068 #include "mtk_curl_2d.h"
00069
00070 mtk::UniStgGrid3D mtk::Curl2D::operator*(const
 mtk::UniStgGrid2D &grid) const {
00071
00072
00073 mtk::UniStgGrid3D output;
00074
00075 return output;
00076 }
00077
00078
00079 mtk::Curl2D::Curl2D():
00080 order_accuracy_(),
00081 mimetic_threshold_() {}
00082
00083 mtk::Curl2D::Curl2D(const Curl2D &curl):
00084 order_accuracy_(curl.order_accuracy_),
00085 mimetic_threshold_(curl.mimetic_threshold_) {}
00086
00087 mtk::Curl2D::~Curl2D() {}

```

```

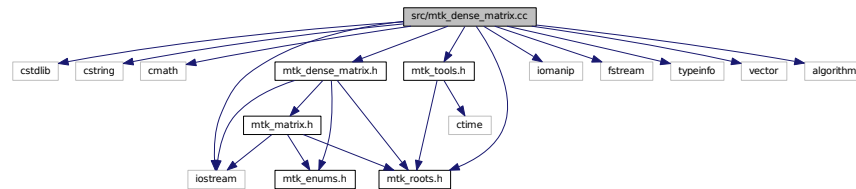
00088
00089 bool mtk::Curl2D::ConstructCurl2D(const
 mtk::UniStgGrid2D &grid,
00090 int order_accuracy,
00091 mtk::Real mimetic_threshold) {
00092
00093 int num_cells_x = grid.num_cells_x();
00094 int num_cells_y = grid.num_cells_y();
00095
00096 int mx = num_cells_x + 2; // Dx vertical dimension.
00097 int nx = num_cells_x + 1; // Dx horizontal dimension.
00098 int my = num_cells_y + 2; // Dy vertical dimension.
00099 int ny = num_cells_y + 1; // Dy horizontal dimension.
00100
00101 mtk::Div1D div;
00102
00103 bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00104
00105 #ifdef MTK_PERFORM_PREVENTIONS
00106 if (!info) {
00107 std::cerr << "Mimetic div could not be built." << std::endl;
00108 return info;
00109 }
00110 #endif
00111
00112 auto west = grid.west_bndy();
00113 auto east = grid.east_bndy();
00114 auto south = grid.south_bndy();
00115 auto north = grid.east_bndy();
00116
00117 mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00118 mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00119
00120 mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00121 mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00122
00123 bool padded{true};
00124 bool transpose{false};
00125
00126 mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00127 mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00128
00129 mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00130 mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00131
00132 #if MTK_VERBOSE_LEVEL > 2
00133 std::cout << "Dx: " << mx << " by " << nx << std::endl;
00134 std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;
00135 std::cout << "Dy: " << my << " by " << ny << std::endl;
00136 std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00137 std::cout << "Curl 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
 nx*ny << std::endl;
00138 #endif
00139
00140 mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00141
00142 for (auto ii = 0; ii < mx*my; ii++) {
00143 for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00144 d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00145 }
00146 for (auto kk=0; kk<ny*num_cells_x; kk++) {
00147 d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00148 }
00149 }
00150
00151 curl_ = d2d;
00152
00153 return info;
00154 }
00155
00156
00157 mtk::DenseMatrix mtk::Curl2D::ReturnAsDenseMatrix() const {
00158
00159 return curl_;
00160 }

```

## 18.81 src/mtk\_dense\_matrix.cc File Reference

```
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <typeinfo>
#include <vector>
#include <algorithm>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_tools.h"
```

Include dependency graph for mtk\_dense\_matrix.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::DenseMatrix &in)`

## 18.82 mtk\_dense\_matrix.cc

```
00001
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
```

```

00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #include <cstdlib>
00060 #include <cstring>
00061 #include <cmath>
00062
00063 #include <iostream>
00064 #include <iomanip>
00065 #include <fstream>
00066
00067 #include <typeinfo>
00068
00069 #include <vector>
00070
00071 #include <algorithm>
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_dense_matrix.h"
00075 #include "mtk_tools.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00080
00081 int mm{in.matrix_properties_.num_rows()}; // Auxiliary.
00082 int nn{in.matrix_properties_.num_cols()}; // Auxiliary.
00083 int output_precision{4};
00084 int output_width{10};
00085
00086 if (in.matrix_properties_.ordering() ==
00087 mtk::MatrixOrdering::COL_MAJOR) {
00088 std::swap(mm, nn);
00089 }
00090 for (int ii = 0; ii < mm; ii++) {
00091 int offset{ii*nn};
00092 for (int jj = 0; jj < nn; jj++) {
00093 mtk::Real value = in.data_[offset + jj];
00094 stream << std::setprecision(output_precision) <<
00095 std::setw(output_width) << value;
00096 }
00097 stream << std::endl;
00098 }
00099 if (in.matrix_properties_.ordering() ==
00100 mtk::MatrixOrdering::COL_MAJOR) {
00101 std::swap(mm, nn);
00102 }
00103 return stream;
00104 }
00105 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
00106 mtk::DenseMatrix &in) {
00107 if(this == &in) {
00108 return *this;
00109 }

```

```

00110
00111 matrix_properties_.set_storage(in.
matrix_properties_.storage());
00112
00113 matrix_properties_.set_ordering(in.
matrix_properties_.ordering());
00114
00115 auto aux = in.matrix_properties_.num_rows();
00116 matrix_properties_.set_num_rows(aux);
00117
00118 aux = in.matrix_properties().num_cols();
00119 matrix_properties_.set_num_cols(aux);
00120
00121 aux = in.matrix_properties().num_zero();
00122 matrix_properties_.set_num_zero(aux);
00123
00124 aux = in.matrix_properties().num_null();
00125 matrix_properties_.set_num_null(aux);
00126
00127 auto num_rows = matrix_properties_.num_rows();
00128 auto num_cols = matrix_properties_.num_cols();
00129
00130 delete [] data_;
00131
00132 try {
00133 data_ = new mtk::Real[num_rows*num_cols];
00134 } catch (std::bad_alloc &memory_allocation_exception) {
00135 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00136 std::endl;
00137 std::cerr << memory_allocation_exception.what() << std::endl;
00138 }
00139 memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
num_cols);
00140
00141 std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00142
00143 return *this;
00144 }
00145
00146 bool mtk::DenseMatrix::operator ==(const
DenseMatrix &in) {
00147
00148 bool ans{true};
00149
00150 auto mm = in.num_rows();
00151 auto nn = in.num_cols();
00152
00153 if (mm != matrix_properties_.num_rows() ||
00154 nn != matrix_properties_.num_cols()) {
00155 return false;
00156 }
00157
00158 for (int ii = 0; ii < mm && ans; ++ii) {
00159 for (int jj = 0; jj < nn && ans; ++jj) {
00160 ans = ans &&
00161 abs(data_[ii*nn + jj] - in.data()[ii*nn + jj]) <
mtk::kDefaultTolerance;
00162 }
00163 }
00164 return ans;
00165 }
00166
00167 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00168
00169 matrix_properties_.set_storage(
mtk::MatrixStorage::DENSE);
00170 matrix_properties_.set_ordering(
mtk::MatrixOrdering::ROW_MAJOR);
00171 }
00172
00173 mtk::DenseMatrix::DenseMatrix(const
mtk::DenseMatrix &in) {
00174
00175 matrix_properties_.set_storage(in.matrix_properties_.storage());
00176
00177 matrix_properties_.set_ordering(in.matrix_properties_.
ordering());
00178
00179 auto aux = in.matrix_properties_.num_rows();
00180 matrix_properties_.set_num_rows(aux);
00181

```



```

00182 aux = in.matrix_properties().num_cols();
00183 matrix_properties_.set_num_cols(aux);
00184
00185 aux = in.matrix_properties().num_zero();
00186 matrix_properties_.set_num_zero(aux);
00187
00188 aux = in.matrix_properties().num_null();
00189 matrix_properties_.set_num_null(aux);
00190
00191 auto num_rows = in.matrix_properties_.num_rows();
00192 auto num_cols = in.matrix_properties_.num_cols();
00193
00194 try {
00195 data_ = new mtk::Real[num_rows*num_cols];
00196 } catch (std::bad_alloc &memory_allocation_exception) {
00197 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00198 std::endl;
00199 std::cerr << memory_allocation_exception.what() << std::endl;
00200 }
00201 memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00202
00203 std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00204 }
00205
00206 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00207
00208 #ifdef MTK_PERFORM_PREVENTIONS
00209 mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00210 mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00211 #endif
00212
00213 matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00214 matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00215 matrix_properties_.set_num_rows(num_rows);
00216 matrix_properties_.set_num_cols(num_cols);
00217
00218 try {
00219 data_ = new mtk::Real[num_rows*num_cols];
00220 } catch (std::bad_alloc &memory_allocation_exception) {
00221 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00222 std::endl;
00223 std::cerr << memory_allocation_exception.what() << std::endl;
00224 }
00225 memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00226 }
00227
00228 mtk::DenseMatrix::DenseMatrix(const int &rank,
00229 const bool &padded,
00230 const bool &transpose) {
00231
00232 #ifdef MTK_PERFORM_PREVENTIONS
00233 mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00234 #endif
00235
00236 int aux{}; // Used to control the padding.
00237
00238 if (padded) {
00239 aux = 1;
00240 }
00241
00242 matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00243 matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00244 matrix_properties_.set_num_rows(aux + rank + aux);
00245 matrix_properties_.set_num_cols(rank);
00246
00247 try {
00248 data_ = new mtk::Real[matrix_properties_.num_values()];
00249 } catch (std::bad_alloc &memory_allocation_exception) {
00250 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00251 std::endl;
00252 std::cerr << memory_allocation_exception.what() << std::endl;
00253 }
00254 memset(data_,
00255 mtk::kZero,
00256 sizeof(data_[0])*(matrix_properties_.num_values()));
00257
00258 for (auto ii = 0; ii < matrix_properties_.num_rows(); ++ii) {
00259 for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00260 data_[ii*matrix_properties_.num_cols() + jj] =
00261 (ii == jj + aux)? mtk::kOne : mtk::kZero;
00262 }
00263 }

```

```

00263 }
00264 if (transpose) {
00265 Transpose();
00266 }
00267 }
00268
00269 mtk::DenseMatrix::DenseMatrix(const mtk::Real *const gen,
00270 const int &gen_length,
00271 const int &pro_length,
00272 const bool &transpose) {
00273
00274 #ifdef MTK_PERFORM_PREVENTIONS
00275 mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00276 mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00277 mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00278 #endif
00279
00280 matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00281 matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00282 if (!transpose) {
00283 matrix_properties_.set_num_rows(gen_length);
00284 matrix_properties_.set_num_cols(pro_length);
00285 } else {
00286 matrix_properties_.set_num_rows(pro_length);
00287 matrix_properties_.set_num_cols(gen_length);
00288 }
00289
00290 int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00291 int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00292
00293 try {
00294 data_ = new mtk::Real[mm*nn];
00295 } catch (std::bad_alloc &memory_allocation_exception) {
00296 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00297 std::endl;
00298 std::cerr << memory_allocation_exception.what() << std::endl;
00299 }
00300 memset(data_, mtk::kZero, sizeof(data_[0])*mm*nn);
00301
00302 if (!transpose) {
00303 for (auto ii = 0; ii < mm; ii++) {
00304 for (auto jj = 0; jj < nn; jj++) {
00305 data_[ii*nn + jj] = pow(gen[ii], (double) jj);
00306 }
00307 }
00308 } else {
00309 for (auto ii = 0; ii < mm; ii++) {
00310 for (auto jj = 0; jj < nn; jj++) {
00311 data_[ii*nn + jj] = pow(gen[jj], (double) ii);
00312 }
00313 }
00314 }
00315 }
00316
00317 mtk::DenseMatrix::~DenseMatrix() {
00318 delete [] data_;
00319 data_ = nullptr;
00320 }
00321
00322
00323 mtk::Matrix mtk::DenseMatrix::matrix_properties() const
00324 noexcept {
00325 return matrix_properties_;
00326 }
00327
00328 void mtk::DenseMatrix::SetOrdering(
00329 mtk::MatrixOrdering oo) noexcept {
00330
00331 #ifdef MTK_PERFORM_PREVENTIONS
00332 mtk::Tools::Prevent(!(oo == mtk::MatrixOrdering::ROW_MAJOR
00333 || oo ==
00334 mtk::MatrixOrdering::COL_MAJOR),
00335 __FILE__, __LINE__, __func__);
00336 #endif
00337 matrix_properties_.set_ordering(oo);
00338 }
00339
00340 int mtk::DenseMatrix::num_rows() const noexcept {

```

```

00341 return matrix_properties_.num_rows();
00342 }
00343
00344 int mtk::DenseMatrix::num_cols() const noexcept {
00345
00346 return matrix_properties_.num_cols();
00347 }
00348
00349 mtk::Real* mtk::DenseMatrix::data() const noexcept {
00350
00351 return data_;
00352 }
00353
00354 mtk::Real mtk::DenseMatrix::GetValue(
00355 const int &mm,
00356 const int &nn) const noexcept {
00357
00358 #ifdef MTK_PERFORM_PREVENTIONS
00359 mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00360 mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00361 #endif
00362
00363 return data_[mm*matrix_properties_.num_cols() + nn];
00364 }
00365
00366 void mtk::DenseMatrix::SetValue(
00367 const int &mm,
00368 const int &nn,
00369 const mtk::Real &val) noexcept {
00370
00371 #ifdef MTK_PERFORM_PREVENTIONS
00372 mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00373 mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00374 #endif
00375
00376 data_[mm*matrix_properties_.num_cols() + nn] = val;
00377 }
00378
00379 void mtk::DenseMatrix::Transpose() {
00380
00381 mtk::Real *data_transposed{}; // Buffer.
00382
00383 int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00384 int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00385
00386 try {
00387 data_transposed = new mtk::Real[mm*nn];
00388 } catch (std::bad_alloc &memory_allocation_exception) {
00389 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00390 std::endl;
00391 std::cerr << memory_allocation_exception.what() << std::endl;
00392 }
00393 memset(data_transposed,
00394 mtk::kZero,
00395 sizeof(data_transposed[0])*mm*nn);
00396
00397 // Assign the values to their transposed position.
00398 for (auto ii = 0; ii < mm; ++ii) {
00399 for (auto jj = 0; jj < nn; ++jj) {
00400 data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00401 }
00402 }
00403
00404 // Swap pointers.
00405 auto tmp = data_; // Temporal holder.
00406 data_ = data_transposed;
00407 delete [] tmp;
00408 tmp = nullptr;
00409
00410 matrix_properties_.set_num_rows(nn);
00411 matrix_properties_.set_num_cols(mm);
00412 }
00413
00414 void mtk::DenseMatrix::OrderRowMajor() {
00415
00416 if (matrix_properties_.ordering() == mtk::MatrixOrdering::COL_MAJOR) {
00417
00418 mtk::Real *data_transposed{}; // Buffer.
00419
00420
00421
00422
00423

```

```

00424 int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00425 int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00426
00427 try {
00428 data_transposed = new mtk::Real[mm*nn];
00429 } catch (std::bad_alloc &memory_allocation_exception) {
00430 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00431 std::endl;
00432 std::cerr << memory_allocation_exception.what() << std::endl;
00433 }
00434 memset(data_transposed,
00435 mtk::kZero,
00436 sizeof(data_transposed[0])*mm*nn);
00437
00438 // Assign the values to their transposed position.
00439 std::swap(mm, nn);
00440 for (auto ii = 0; ii < mm; ++ii) {
00441 for (auto jj = 0; jj < nn; ++jj) {
00442 data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00443 }
00444 }
00445 std::swap(mm, nn);
00446
00447 // Swap pointers.
00448 auto tmp = data_; // Temporal holder.
00449 data_ = data_transposed;
00450 delete [] tmp;
00451 tmp = nullptr;
00452
00453 matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00454 }
00455 }
00456
00457 void mtk::DenseMatrix::OrderColMajor() {
00458
00459 if (matrix_properties_.ordering() == ROW_MAJOR) {
00460
00461 mtk::Real *data_transposed{}; // Buffer.
00462
00463 int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00464 int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00465
00466 try {
00467 data_transposed = new mtk::Real[mm*nn];
00468 } catch (std::bad_alloc &memory_allocation_exception) {
00469 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00470 std::endl;
00471 std::cerr << memory_allocation_exception.what() << std::endl;
00472 }
00473 memset(data_transposed,
00474 mtk::kZero,
00475 sizeof(data_transposed[0])*mm*nn);
00476
00477 // Assign the values to their transposed position.
00478 for (auto ii = 0; ii < mm; ++ii) {
00479 for (auto jj = 0; jj < nn; ++jj) {
00480 data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00481 }
00482 }
00483 }
00484
00485 // Swap pointers.
00486 auto tmp = data_; // Temporal holder.
00487 data_ = data_transposed;
00488 delete [] tmp;
00489 tmp = nullptr;
00490
00491 matrix_properties_.set_ordering(mtk::MatrixOrdering::COL_MAJOR);
00492 }
00493 }
00494 }
00495
00496 mtk::DenseMatrix mtk::DenseMatrix::Kron(const
 mtk::DenseMatrix &aa,
00497 const mtk::DenseMatrix &bb) {
00498
00499 int row_offset{}; // Offset for rows.
00500 int col_offset{}; // Offset for rows.
00501
00502 mtk::Real aa_factor{}; // Used in computation.
00503
00504
00505

```

```

00506 // Auxiliary variables:
00507 auto aux1 = aa.matrix_properties_.num_rows()*bb.
matrix_properties_.num_rows();
00508 auto aux2 = aa.matrix_properties_.num_cols()*bb.
matrix_properties_.num_cols();
00509
00510 mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00511
00512 int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00513
00514 auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00515 auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00516 auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00517 auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00518
00519 for (auto ii = 0; ii < mm; ++ii) {
00520 row_offset = ii*pp;
00521 for (auto jj = 0; jj < nn; ++jj) {
00522 col_offset = jj*qq;
00523 aa_factor = aa.data_[ii*nn + jj];
00524 for (auto ll = 0; ll < pp; ++ll) {
00525 for (auto oo = 0; oo < qq; ++oo) {
00526 auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00527 output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00528 }
00529 }
00530 }
00531 }
00532
00533 output.matrix_properties_.set_storage(
mtk::MatrixStorage::DENSE);
00534 output.matrix_properties_.set_ordering(
mtk::MatrixOrdering::ROW_MAJOR);
00535
00536 return output;
00537 }
00538
00539 bool mtk::DenseMatrix::WriteToFile(const std::string &filename) const {
00540
00541 std::ofstream output_dat_file; // Output file.
00542
00543 output_dat_file.open(filename);
00544
00545 if (!output_dat_file.is_open()) {
00546 return false;
00547 }
00548
00549 int mm{matrix_properties_.num_rows()};
00550 int nn{matrix_properties_.num_cols()};
00551
00552 for (int ii = 0; ii < mm; ++ii) {
00553 int offset{ii*nn};
00554 for (int jj = 0; jj < nn; ++jj) {
00555 output_dat_file << ii << ' ' << jj << ' ' << data_[offset + jj] <<
std::endl;
00556 }
00557 }
00558
00559 output_dat_file.close();
00560
00561 return true;
00562 }
00563 }

```

## 18.83 src/mtk\_div\_1d.cc File Reference

Implements the class Div1D.



## 18.84 mtk\_div\_1d.cc

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075
00076 #include "mtk_blas_adapter.h"
00077 #include "mtk_lapack_adapter.h"
00078 #include "mtk_glpk_adapter.h"
00079
00080 #include "mtk_div_1d.h"
00081
00082 namespace mtk {
00083
00084 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00085
00086 int output_precision{5};
00087 int output_width{8};
00088
00089 stream << "divergence_[0] = " << std::setprecision(output_precision) <<
00090 std::setw(output_width) << in.divergence_[0] <<

```

```

00093 std::endl;
00094
00095
00096
00097 stream << "divergence_[" << in.order_accuracy_ << "] = ";
00098 for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00099 stream << std::setprecision(output_precision) <<
00100 std::setw(output_width) << in.divergence_[ii] << " ";
00101 }
00102 stream << std::endl;
00103
00104 if (in.order_accuracy_ > 2) {
00105
00106
00107 stream << "divergence_[" << in.order_accuracy_ + 1 << ":" <<
00108 2*in.order_accuracy_ << "] = ";
00109 for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00110 order_accuracy_; ++ii) {
00111 stream << std::setprecision(output_precision) <<
00112 std::setw(output_width) << in.divergence_[ii] << " ";
00113 }
00114 stream << std::endl;
00115
00116
00117 auto offset = (2*in.order_accuracy_ + 1);
00118 int mm{};
00119 for (auto ii = 0; ii < in.dim_null_; ++ii) {
00120 stream << "divergence_[" << offset + mm << ":" <<
00121 offset + mm + in.num_bndy_coeffs_ - 1 << "] = ";
00122 for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {
00123 auto value = in.divergence_[offset + mm];
00124 stream << std::setprecision(output_precision) <<
00125 std::setw(output_width) << value << " ";
00126 ++mm;
00127 }
00128 stream << std::endl;
00129 }
00130 }
00131 }
00132
00133 return stream;
00134 }
00135 }
00136
00137 mtk::Div1D::Div1D():
00138 order_accuracy_(mtk::kDefaultOrderAccuracy),
00139 dim_null_(),
00140 num_bndy_coeffs_(),
00141 divergence_length_(),
00142 minrow_(),
00143 row_(),
00144 coeffs_interior_(),
00145 prem_apps_(),
00146 weights_crs_(),
00147 weights_cbs_(),
00148 mim_bndy_(),
00149 divergence_(),
00150 mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00151
00152 mtk::Div1D::Div1D(const Div1D &div):
00153 order_accuracy_(div.order_accuracy_),
00154 dim_null_(div.dim_null_),
00155 num_bndy_coeffs_(div.num_bndy_coeffs_),
00156 divergence_length_(div.divergence_length_),
00157 minrow_(div.minrow_),
00158 row_(div.row_),
00159 coeffs_interior_(div.coeffs_interior_),
00160 prem_apps_(div.prem_apps_),
00161 weights_crs_(div.weights_crs_),
00162 weights_cbs_(div.weights_cbs_),
00163 mim_bndy_(div.mim_bndy_),
00164 divergence_(div.divergence_),
00165 mimetic_threshold_(div.mimetic_threshold_) {}
00166
00167 mtk::Div1D::~~Div1D() {
00168
00169 delete[] coeffs_interior_;
00170 coeffs_interior_ = nullptr;
00171
00172 delete[] prem_apps_;
00173 prem_apps_ = nullptr;
00174
00175 delete[] weights_crs_;

```



```

00176 weights_crs_ = nullptr;
00177
00178 delete[] weights_cbs_;
00179 weights_cbs_ = nullptr;
00180
00181 delete[] mim_bndy_;
00182 mim_bndy_ = nullptr;
00183
00184 delete[] divergence_;
00185 divergence_ = nullptr;
00186 }
00187
00188 bool mtk::Div1D::ConstructDiv1D(int order_accuracy,
00189 mtk::Real mimetic_threshold) {
00190
00191 #ifdef MTK_PERFORM_PREVENTIONS
00192 mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00193 mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00194 mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00195 __FILE__, __LINE__, __func__);
00196
00197 if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00198 std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00199 }
00200
00201 std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00202 std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00203 #endif
00204
00205 order_accuracy_ = order_accuracy;
00206 mimetic_threshold_ = mimetic_threshold;
00207
00208
00209 bool abort_construction = ComputeStencilInteriorGrid();
00210
00211 #ifdef MTK_PERFORM_PREVENTIONS
00212 if (!abort_construction) {
00213 std::cerr << "Could NOT complete stage 1." << std::endl;
00214 std::cerr << "Exiting..." << std::endl;
00215 return false;
00216 }
00217 #endif
00218
00219 // At this point, we already have the values for the interior stencil stored
00220 // in the coeffs_interior_ array.
00221
00222 // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00223 // approximation at the boundary, thus it has no weights. For this case, the
00224 // dimension of the null-space of the Vandermonde matrices used to compute the
00225 // approximating coefficients at the boundary is 0. Ergo, we compute this
00226 // number first and then decide if we must compute anything at the boundary.
00227
00228 dim_null_ = order_accuracy_/2 - 1;
00229
00230 if (dim_null_ > 0) {
00231
00232 #ifdef MTK_PRECISION_DOUBLE
00233 num_bndy_coeffs_ = (int) (3.0*(mtk::Real) order_accuracy_)/2.0);
00234 #else
00235 num_bndy_coeffs_ = (int) (3.0f*(mtk::Real) order_accuracy_)/2.0f);
00236 #endif
00237
00238
00239 // For this we will follow recommendations given in:
00240 //
00241 // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00242 //
00243 // We will compute the QR Factorization of the transpose, as in the
00244 // following (MATLAB) pseudo-code:
00245 //
00246 // [Q,R] = qr(V'); % Full QR as defined in
00247 // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00248 //
00249 // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q);
00250 //
00251 // However, given the nature of the Vandermonde matrices we've just
00252 // computed, they all posses the same null-space. Therefore, we impose the
00253 // convention of computing the null-space of the first Vandermonde matrix
00254 // (west boundary).
00255
00256 abort_construction = ComputeRationalBasisNullSpace();
00257
00258

```

```

00259
00260 #ifdef MTK_PERFORM_PREVENTIONS
00261 if (!abort_construction) {
00262 std::cerr << "Could NOT complete stage 2.1." << std::endl;
00263 std::cerr << "Exiting..." << std::endl;
00264 return false;
00265 }
00266 #endif
00267
00269 abort_construction = ComputePreliminaryApproximations();
00270
00271
00272 #ifdef MTK_PERFORM_PREVENTIONS
00273 if (!abort_construction) {
00274 std::cerr << "Could NOT complete stage 2.2." << std::endl;
00275 std::cerr << "Exiting..." << std::endl;
00276 return false;
00277 }
00278 #endif
00279
00281 abort_construction = ComputeWeights();
00282
00283
00284 #ifdef MTK_PERFORM_PREVENTIONS
00285 if (!abort_construction) {
00286 std::cerr << "Could NOT complete stage 2.3." << std::endl;
00287 std::cerr << "Exiting..." << std::endl;
00288 return false;
00289 }
00290 #endif
00291
00293 abort_construction = ComputeStencilBoundaryGrid();
00294
00295
00296 #ifdef MTK_PERFORM_PREVENTIONS
00297 if (!abort_construction) {
00298 std::cerr << "Could NOT complete stage 2.4." << std::endl;
00299 std::cerr << "Exiting..." << std::endl;
00300 return false;
00301 }
00302 #endif
00303
00304 } // End of: if (dim_null_ > 0);
00305
00307
00308 // Once we have the following three collections of data:
00309 // (a) the coefficients for the interior,
00310 // (b) the coefficients for the boundary (if it applies),
00311 // (c) and the weights (if it applies),
00312 // we will store everything in the output array:
00313
00314 abort_construction = AssembleOperator();
00315
00316 #ifdef MTK_PERFORM_PREVENTIONS
00317 if (!abort_construction) {
00318 std::cerr << "Could NOT complete stage 3." << std::endl;
00319 std::cerr << "Exiting..." << std::endl;
00320 return false;
00321 }
00322 #endif
00323
00324 return true;
00325 }
00326
00327 int mtk::Div1D::num_bndy_coeffs() const {
00328
00329 return num_bndy_coeffs_;
00330 }
00331
00332 mtk::Real *mtk::Div1D::coeffs_interior() const {
00333
00334 return coeffs_interior_;
00335 }
00336
00337 mtk::Real *mtk::Div1D::weights_crs() const {
00338
00339 return weights_crs_;
00340 }
00341
00342 mtk::Real *mtk::Div1D::weights_cbs() const {
00343

```

```

00344 return weights_cbs_;
00345 }
00346
00347 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00348
00349 mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00350
00351 auto counter = 0;
00352 for (auto ii = 0; ii < dim_null_; ++ii) {
00353 for(auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00354 xx.SetValue(ii,jj, divergence_[2*order_accuracy_ + 1 + counter]);
00355 counter++;
00356 }
00357 }
00358
00359 return xx;
00360 }
00361
00362 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(
00363 const UniStgGrid1D &grid) const {
00364
00365 int nn{grid.num_cells_x()}; // Number of cells on the grid.
00366
00367 #ifdef MTK_PERFORM_PREVENTIONS
00368 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00369 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00370 #endif
00371
00372 mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00373
00374 int dd_num_rows = nn + 2;
00375 int dd_num_cols = nn + 1;
00376 int elements_per_row = num_bndy_coeffs_;
00377 int num_extra_rows = dim_null_;
00378
00379 // Output matrix featuring sizes for divergence operators.
00380 mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00381
00382
00383
00384 auto ee_index = 0;
00385 for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00386 auto cc = 0;
00387 for(auto jj = 0 ; jj < dd_num_rows; jj++) {
00388 if(cc >= elements_per_row) {
00389 out.SetValue(ii, jj, mtk::kZero);
00390 } else {
00391 out.SetValue(ii, jj, mim_bndy_[ee_index++]*inv_delta_x);
00392 cc++;
00393 }
00394 }
00395 }
00396
00397
00398
00399 for (auto ii = num_extra_rows + 1;
00400 ii < dd_num_rows - num_extra_rows - 1; ii++) {
00401 auto jj = ii - num_extra_rows - 1;
00402 for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00403 out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00404 }
00405 }
00406
00407
00408
00409 ee_index = 0;
00410 for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--) {
00411 {
00412 auto cc = 0;
00413 for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00414 if(cc >= elements_per_row) {
00415 out.SetValue(ii,jj,0.0);
00416 } else {
00417 out.SetValue(ii, jj, -mim_bndy_[ee_index++]*inv_delta_x);
00418 cc++;
00419 }
00420 }
00421 }
00422
00423 return out;
00424 }
00425
00426 mtk::DenseMatrix mtk::Div1D::ReturnAsDimensionlessDenseMatrix
00427 (

```

```

00427 int num_cells_x) const {
00428
00429 int nn{num_cells_x}; // Number of cells on the grid.
00430
00431 #ifdef MTK_PERFORM_PREVENTIONS
00432 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00433 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00434 #endif
00435
00436 int dd_num_rows = nn + 2;
00437 int dd_num_cols = nn + 1;
00438 int elements_per_row = num_bndy_coeffs_;
00439 int num_extra_rows = dim_null_;
00440
00441 // Output matrix featuring sizes for gradient operators.
00442 mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00443
00444
00445 auto ee_index = 0;
00446 for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00447 auto cc = 0;
00448 for(auto jj = 0 ; jj < dd_num_rows; jj++) {
00449 if(cc >= elements_per_row) {
00450 out.SetValue(ii, jj, mtk::kZero);
00451 } else {
00452 out.SetValue(ii, jj, mim_bndy_[ee_index++]);
00453 cc++;
00454 }
00455 }
00456 }
00457 }
00458
00459 for (auto ii = num_extra_rows + 1;
00460 ii < dd_num_rows - num_extra_rows - 1; ii++) {
00461 auto jj = ii - num_extra_rows - 1;
00462 for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00463 out.SetValue(ii, jj, coeffs_interior_[cc]);
00464 }
00465 }
00466
00467 ee_index = 0;
00468 for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--)
00469 {
00470 auto cc = 0;
00471 for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00472 if(cc >= elements_per_row) {
00473 out.SetValue(ii, jj, 0.0);
00474 } else {
00475 out.SetValue(ii, jj, -mim_bndy_[ee_index++]);
00476 cc++;
00477 }
00478 }
00479 }
00480
00481 return out;
00482 }
00483
00484 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00485
00486 mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00487
00488 try {
00489 pp = new mtk::Real[order_accuracy_];
00490 } catch (std::bad_alloc &memory_allocation_exception) {
00491 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00492 std::endl;
00493 std::cerr << memory_allocation_exception.what() << std::endl;
00494 }
00495 memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00496
00497 #ifdef MTK_PRECISION_DOUBLE
00498 pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00499 #else
00500 pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00501 #endif
00502
00503 for (auto ii = 1; ii < order_accuracy_; ++ii) {
00504 pp[ii] = pp[ii - 1] + mtk::kOne;
00505 }

```

```

00512
00513 #if MTK_VERBOSE_LEVEL > 3
00514 std::cout << "pp =" << std::endl;
00515 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00516 std::cout << std::setw(12) << pp[ii];
00517 }
00518 std::cout << std::endl << std::endl;
00519 #endif
00520
00522
00523 bool transpose{false};
00524
00525 mtk::DenseMatrix vander_matrix(pp,
00526 order_accuracy_,
00527 order_accuracy_,
00528 transpose);
00529
00530 #if MTK_VERBOSE_LEVEL > 4
00531 std::cout << "vander_matrix = " << std::endl;
00532 std::cout << vander_matrix << std::endl;
00533 #endif
00534
00536
00537 try {
00538 coeffs_interior_ = new mtk::Real[order_accuracy_];
00539 } catch (std::bad_alloc &memory_allocation_exception) {
00540 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00541 std::endl;
00542 std::cerr << memory_allocation_exception.what() << std::endl;
00543 }
00544 memset(coeffs_interior_,
00545 mtk::kZero,
00546 sizeof(coeffs_interior_[0])*order_accuracy_);
00547
00548 coeffs_interior_[1] = mtk::kOne;
00549
00550 #if MTK_VERBOSE_LEVEL > 3
00551 std::cout << "oo =" << std::endl;
00552 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00553 std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00554 }
00555 std::cout << std::endl;
00556 #endif
00557
00559
00560 int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00561 coeffs_interior_)};
00562
00563 #ifdef MTK_PERFORM_PREVENTIONS
00564 if (!info) {
00565 std::cout << "System solved! Interior stencil attained!" << std::endl;
00566 std::cout << std::endl;
00567 }
00568 else {
00569 std::cerr << "Something wrong solving system! info = " << info << std::endl;
00570 std::cerr << "Exiting..." << std::endl;
00571 return false;
00572 }
00573 #endif
00574
00575 #if MTK_VERBOSE_LEVEL > 3
00576 std::cout << "coeffs_interior_ =" << std::endl;
00577 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00578 std::cout << std::setw(12) << coeffs_interior_[ii];
00579 }
00580 std::cout << std::endl << std::endl;
00581 #endif
00582
00583 delete [] pp;
00584 pp = nullptr;
00585
00586 return true;
00587 }
00588
00589 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
00590
00591 mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00592
00594
00595 try {
00596 gg = new mtk::Real[num_bndy_coeffs_];

```

```

00597 } catch (std::bad_alloc &memory_allocation_exception) {
00598 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00599 std::endl;
00600 std::cerr << memory_allocation_exception.what() << std::endl;
00601 }
00602 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00603
00604 #ifdef MTK_PRECISION_DOUBLE
00605 gg[0] = -1.0/2.0;
00606 #else
00607 gg[0] = -1.0f/2.0f;
00608 #endif
00609 for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00610 gg[ii] = gg[ii - 1] + mtk::kOne;
00611 }
00612
00613 #if MTK_VERBOSE_LEVEL > 3
00614 std::cout << "gg =" << std::endl;
00615 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00616 std::cout << std::setw(12) << gg[ii];
00617 }
00618 std::cout << std::endl << std::endl;
00619 #endif
00620
00622 bool tran{true}; // Should I transpose the Vandermonde matrix.
00623
00624 mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00625
00626 #if MTK_VERBOSE_LEVEL > 4
00627 std::cout << "vv_west_t =" << std::endl;
00628 std::cout << vv_west_t << std::endl;
00629 #endif
00630
00631 mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00632 (vv_west_t));
00633
00634 #if MTK_VERBOSE_LEVEL > 4
00635 std::cout << "QQ^T =" << std::endl;
00636 std::cout << qq_t << std::endl;
00637 #endif
00638
00640 int KK_num_rows_{num_bndy_coeffs_};
00641 int KK_num_cols_{dim_null_};
00642
00643 mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00644
00645 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00646 for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00647 KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00648 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00649 }
00650 }
00651
00652 #if MTK_VERBOSE_LEVEL > 2
00653 std::cout << "KK =" << std::endl;
00654 std::cout << KK << std::endl;
00655 std::cout << "KK.num_rows() = " << KK.num_rows() << std::endl;
00656 std::cout << "KK.num_cols() = " << KK.num_cols() << std::endl;
00657 #endif
00658
00659 // Scale thus requesting that the last entries of the attained basis for the
00660 // null-space, adopt the pattern we require.
00661 // Essentially we will implement the following MATLAB pseudo-code:
00662 // scalers = KK(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:) \ B
00663 // SK = KK*scalers
00664 // where SK is the scaled null-space.
00665
00666 // In this point, we almost have all the data we need correctly allocated
00667 // in memory. We will create the matrix II_, and elements we wish to scale in
00668 // the KK array. Using the concept of the leading dimension, we could just
00669 // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00670 // GET how does it work. So I will just create a matrix with the content of
00671 // this array that we need, solve for the scalers and then scale the
00672 // whole KK:
00673
00674 // We will then create memory for that sub-matrix of KK (SUBK).

```

```

00681
00682 mtk::DenseMatrix SUBK(dim_null_, dim_null_);
00683
00684 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00685 for (auto jj = 0; jj < dim_null_; ++jj) {
00686 SUBK.data()[ii - (num_bndy_coeffs_ - dim_null_)*dim_null_ + jj] =
00687 KK.data()[ii*dim_null_ + jj];
00688 }
00689 }
00690
00691 #if MTK_VERBOSE_LEVEL > 4
00692 std::cout << "SUBK =" << std::endl;
00693 std::cout << SUBK << std::endl;
00694 #endif
00695
00696 SUBK.Transpose();
00697
00698 #if MTK_VERBOSE_LEVEL > 4
00699 std::cout << "SUBK^T =" << std::endl;
00700 std::cout << SUBK << std::endl;
00701 #endif
00702
00703 bool padded{false};
00704 tran = false;
00705
00706 mtk::DenseMatrix II(dim_null_, padded, tran);
00707
00708 #if MTK_VERBOSE_LEVEL > 4
00709 std::cout << "II =" << std::endl;
00710 std::cout << II << std::endl;
00711 #endif
00712
00713 // Solve the system to compute the scalars.
00714 // An example of the system to solve, for k = 8, is:
00715 //
00716 // SUBK*scalars = II_or
00717 //
00718 // | 0.386018 -0.0339244 -0.129478 | | 1 0 0 |
00719 // | -0.119774 0.0199423 0.0558632 |*scalars = | 0 1 0 |
00720 // | 0.0155708 -0.00349546 -0.00853182 | | 0 0 1 |
00721 //
00722 // Notice this is a nrhs = 3 system.
00723 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalars... they
00724 // will be stored in the created identity matrix.
00725 // Let us first transpose SUBK (because of LAPACK):
00726
00727 int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00728
00729 #ifdef MTK_PERFORM_PREVENTIONS
00730 if (!info) {
00731 std::cout << "System successfully solved!" <<
00732 std::endl;
00733 } else {
00734 std::cerr << "Something went wrong solving system! info = " << info <<
00735 std::endl;
00736 std::cerr << "Exiting..." << std::endl;
00737 return false;
00738 }
00739 std::cout << std::endl;
00740 #endif
00741
00742 #if MTK_VERBOSE_LEVEL > 4
00743 std::cout << "Computed scalars:" << std::endl;
00744 std::cout << II << std::endl;
00745 #endif
00746
00747 // Multiply the two matrices to attain a scaled basis for null-space.
00748
00749 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00750
00751 #if MTK_VERBOSE_LEVEL > 4
00752 std::cout << "Rational basis for the null-space:" << std::endl;
00753 std::cout << rat_basis_null_space_ << std::endl;
00754 #endif
00755
00756 // At this point, we have a rational basis for the null-space, with the
00757 // pattern we need! :)
00758
00759 delete [] gg;
00760 gg = nullptr;
00761

```

```

00762 return true;
00763 }
00764
00765 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00766
00768 mtk::Real *gg{}; // Generator vector for the first approximation.
00770
00771 try {
00772 gg = new mtk::Real[num_bndy_coeffs_];
00773 } catch (std::bad_alloc &memory_allocation_exception) {
00774 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00775 std::endl;
00776 std::cerr << memory_allocation_exception.what() << std::endl;
00777 }
00778 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00779
00780 #ifdef MTK_PRECISION_DOUBLE
00781 gg[0] = -1.0/2.0;
00782 #else
00783 gg[0] = -1.0f/2.0f;
00784 #endif
00785 for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00786 gg[ii] = gg[ii - 1] + mtk::kOne;
00787 }
00788
00789 #if MTK_VERBOSE_LEVEL > 3
00790 std::cout << "gg0 =" << std::endl;
00791 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00792 std::cout << std::setw(12) << gg[ii];
00793 }
00794 std::cout << std::endl << std::endl;
00795 #endif
00796
00797 // Allocate 2D array to store the collection of preliminary approximations.
00798 try {
00799 prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00800 } catch (std::bad_alloc &memory_allocation_exception) {
00801 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00802 std::endl;
00803 std::cerr << memory_allocation_exception.what() << std::endl;
00804 }
00805 memset(prem_apps_,
00806 mtk::kZero,
00807 sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00808
00810
00811 for (auto ll = 0; ll < dim_null_; ++ll) {
00812
00813 // Re-check new generator vector for every iteration except for the first.
00814 #if MTK_VERBOSE_LEVEL > 3
00815 if (ll > 0) {
00816 std::cout << "gg" << ll << " =" << std::endl;
00817 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00818 std::cout << std::setw(12) << gg[ii];
00819 }
00820 std::cout << std::endl << std::endl;
00821 }
00822 #endif
00823
00825
00826 bool transpose{false};
00827
00828 mtk::DenseMatrix AA_(gg,
00829 num_bndy_coeffs_, order_accuracy_ + 1,
00830 transpose);
00831
00832 #if MTK_VERBOSE_LEVEL > 4
00833 std::cout << "AA_" << ll << " =" << std::endl;
00834 std::cout << AA_ << std::endl;
00835 #endif
00836
00838
00839 mtk::Real *ob{};
00840
00841 auto ob_ld = num_bndy_coeffs_;
00842
00843 try {
00844 ob = new mtk::Real[ob_ld];
00845 } catch (std::bad_alloc &memory_allocation_exception) {
00846 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```



```

00847 std::endl;
00848 std::cerr << memory_allocation_exception.what() << std::endl;
00849 }
00850 memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00851
00852 ob[1] = mtk::kOne;
00853
00854 #if MTK_VERBOSE_LEVEL > 4
00855 std::cout << "ob = " << std::endl << std::endl;
00856 for (auto ii = 0; ii < ob_ld; ++ii) {
00857 std::cout << std::setw(12) << ob[ii] << std::endl;
00858 }
00859 std::cout << std::endl;
00860 #endif
00861
00862 // However, this is an under-determined system of equations. So we can not
00863 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00864 // our LAPACKAdapter class.
00865
00866 int info_{
00867 mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00868 ob, ob_ld)};
00869
00870 #ifdef MTK_PERFORM_PREVENTIONS
00871 if (!info_) {
00872 std::cout << "System successfully solved!" << std::endl << std::endl;
00873 } else {
00874 std::cerr << "Error solving system! info = " << info_ << std::endl;
00875 }
00876 #endif
00877
00878 #if MTK_VERBOSE_LEVEL > 3
00879 std::cout << "ob =" << std::endl;
00880 for (auto ii = 0; ii < ob_ld; ++ii) {
00881 std::cout << std::setw(12) << ob[ii] << std::endl;
00882 }
00883 std::cout << std::endl;
00884 #endif
00885
00886 // This implies a DAXPY operation. However, we must construct the arguments
00887 // for this operation.
00888
00889 // Save them into the ob_bottom array:
00890
00891 Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00892
00893 try {
00894 ob_bottom = new mtk::Real[dim_null_];
00895 } catch (std::bad_alloc &memory_allocation_exception) {
00896 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00897 std::endl;
00898 std::cerr << memory_allocation_exception.what() << std::endl;
00899 }
00900 memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00901
00902 for (auto ii = 0; ii < dim_null_; ++ii) {
00903 ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00904 }
00905
00906 #if MTK_VERBOSE_LEVEL > 3
00907 std::cout << "ob_bottom =" << std::endl;
00908 for (auto ii = 0; ii < dim_null_; ++ii) {
00909 std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00910 }
00911 std::cout << std::endl;
00912 #endif
00913
00914 // We must computed an scaled ob, sob, using the scaled null-space in
00915 // rat_basis_null_space_.
00916 // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00917 // or:
00918 // ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00919 // thus:
00920 // Y = a*A *x + b*Y (DAXPY).
00921
00922 #if MTK_VERBOSE_LEVEL > 3
00923 std::cout << "Rational basis for the null-space:" << std::endl;
00924 std::cout << rat_basis_null_space_ << std::endl;
00925 #endif
00926
00927
00928
00929
00930

```

```

00931 mtk::Real alpha{-mtk::kOne};
00932 mtk::Real beta{mtk::kOne};
00933
00934 mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00935 ob_bottom, beta, ob);
00936
00937 #if MTK_VERBOSE_LEVEL > 3
00938 std::cout << "scaled ob:" << std::endl;
00939 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00940 std::cout << std::setw(12) << ob[ii] << std::endl;
00941 }
00942 std::cout << std::endl;
00943 #endif
00944
00945 // We save the recently scaled solution, into an array containing these.
00946 // We can NOT start building the pi matrix, simply because I want that part
00947 // to be separated since its construction depends on the algorithm we want
00948 // to implement.
00949
00950 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00951 prem_apps_[ii*dim_null_ + 11] = ob[ii];
00952 }
00953
00954 // After the first iteration, simply shift the entries of the last
00955 // generator vector used:
00956 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00957 gg[ii]--;
00958 }
00959
00960 // Garbage collection for this loop:
00961 delete[] ob;
00962 ob = nullptr;
00963
00964 delete[] ob_bottom;
00965 ob_bottom = nullptr;
00966 } // End of: for (ll = 0; ll < dim_null; ll++);
00967
00968 #if MTK_VERBOSE_LEVEL > 4
00969 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00970 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00971 for (auto jj = 0; jj < dim_null_; ++jj) {
00972 std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00973 }
00974 std::cout << std::endl;
00975 }
00976 std::cout << std::endl;
00977 #endif
00978
00979 delete[] gg;
00980 gg = nullptr;
00981
00982 return true;
00983 }
00984
00985 bool mtk::Div1D::ComputeWeights(void) {
00986
00987 // Matrix to compute the weights as in the CRSA.
00988 mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00989
00990
00991 // Assemble the pi matrix using:
00992 // 1. The collection of scaled preliminary approximations.
00993 // 2. The collection of coefficients approximating at the interior.
00994 // 3. The scaled basis for the null-space.
00995
00996 // 1.1. Process array of scaled preliminary approximations.
00997
00998 // These are queued in scaled_solutions. Each one of these, will be a column
00999 // of the pi matrix:
01000 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01001 for (auto jj = 0; jj < dim_null_; ++jj) {
01002 pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01003 prem_apps_[ii*dim_null_ + jj];
01004 }
01005 }
01006
01007 // 1.2. Add columns from known stencil approximating at the interior.
01008
01009 // However, these must be padded by zeros, according to their position in the
01010 // final pi matrix:
01011 auto mm = 0;

```

```

01013 for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
01014 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01015 pi.data()[(ii + mm)*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01016 coeffs_interior_[ii];
01017 }
01018 ++mm;
01019 }
01020
01021 rat_basis_null_space_.OrderColMajor();
01022
01023 #if MTK_VERBOSE_LEVEL > 4
01024 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01025 std::cout << rat_basis_null_space_ << std::endl;
01026 #endif
01027
01028 // 1.3. Add final set of columns: rational basis for null-space.
01029 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01030 jj < num_bndy_coeffs_ - 1;
01031 ++jj) {
01032 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01033 auto og =
01034 (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01035 auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01036 pi.data()[de] = rat_basis_null_space_.data()[og];
01037 }
01038 }
01039
01040 #if MTK_VERBOSE_LEVEL > 3
01041 std::cout << "coeffs_interior_ =" << std::endl;
01042 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01043 std::cout << std::setw(12) << coeffs_interior_[ii];
01044 }
01045 std::cout << std::endl << std::endl;
01046 #endif
01047
01048 #if MTK_VERBOSE_LEVEL > 4
01049 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01050 std::cout << pi << std::endl;
01051 #endif
01052
01053 // This imposes the mimetic condition.
01054
01055 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01056
01057 try {
01058 hh = new mtk::Real[num_bndy_coeffs_];
01059 } catch (std::bad_alloc &memory_allocation_exception) {
01060 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01061 std::endl;
01062 std::cerr << memory_allocation_exception.what() << std::endl;
01063 }
01064 memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01065
01066 hh[0] = -mtk::kOne;
01067 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01068 auto aux_xx = mtk::kZero;
01069 for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01070 aux_xx += coeffs_interior_[jj];
01071 }
01072 hh[ii] = -mtk::kOne*aux_xx;
01073 }
01074
01075 // That is, we construct a system, to solve for the weights.
01076
01077 // Once again we face the challenge of solving with LAPACK. However, for the
01078 // CRSA, this matrix PI is over-determined, since it has more rows than
01079 // unknowns. However, according to the theory, the solution to this system is
01080 // unique. We will use dgels_.
01081
01082 try {
01083 weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01084 } catch (std::bad_alloc &memory_allocation_exception) {
01085 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01086 std::endl;
01087 std::cerr << memory_allocation_exception.what() << std::endl;
01088 }
01089 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01090
01091 int weights_ld{pi.num_cols() + 1};

```

```

01096
01097 // Preserve hh.
01098 std::copy(hh, hh + weights_ld, weights_cbs_);
01099
01100 pi.Transpose();
01101
01102 int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
01103 pi,
01104 weights_cbs_,
01105 weights_ld)};
01106
01107 #ifdef MTK_PERFORM_PREVENTIONS
01108 if (!info) {
01109 std::cout << "System successfully solved!" << std::endl << std::endl;
01110 } else {
01111 std::cerr << "Error solving system! info = " << info << std::endl;
01112 }
01113 #endif
01114
01115 #if MTK_VERBOSE_LEVEL > 3
01116 std::cout << "hh =" << std::endl;
01117 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01118 std::cout << std::setw(11) << hh[ii] << std::endl;
01119 }
01120 std::cout << std::endl;
01121 #endif
01122 // Preserve the original weights for research.
01123
01124 try {
01125 weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01126 } catch (std::bad_alloc &memory_allocation_exception) {
01127 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01128 std::endl;
01129 std::cerr << memory_allocation_exception.what() << std::endl;
01130 }
01131 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01132
01133 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01134
01135 #if MTK_VERBOSE_LEVEL > 3
01136 std::cout << "weights_CRSA + lambda =" << std::endl;
01137 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01138 std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01139 }
01140 std::cout << std::endl;
01141 #endif
01142
01143 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01144
01145 mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01146
01147 for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01148 for (auto jj = 0; jj < dim_null_; ++jj) {
01149 phi.data()[ii*(order_accuracy_ + 1) + jj] = prem_apps_[ii*dim_null_ + jj];
01150 }
01151 }
01152
01153 int aux{}; // Auxiliary variable.
01154 for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01155 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01156 phi.data()[ii*(order_accuracy_ + 1) + jj] = coeffs_interior[ii];
01157 }
01158 ++aux;
01159 }
01160
01161 for (auto jj = order_accuracy_ - 1; jj >= order_accuracy_ - dim_null_; jj--) {
01162 for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01163 phi.data()[ii*(order_accuracy_ + 1) + jj] = mtk::kZero;
01164 }
01165 }
01166
01167 for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01168 for (auto ii = 0; ii < dim_null_; ++ii) {
01169 phi.data()[ii*(order_accuracy_ + 1) + jj] = -pre_apps_[dim_null_ - ii - 1 + jj*dim_null_];
01170 }
01171 }
01172 }
01173
01174 }
01175
01176 }
01177

```

```

01178 for(auto ii = 0; ii < order_accuracy_/2; ++ii) {
01179 for (auto jj = dim_null_ + 2; jj < order_accuracy_; ++jj) {
01180 auto swap = phi.data()[ii*order_accuracy_+jj];
01181 phi.data()[ii*order_accuracy_ + jj] =
01182 phi.data()[(order_accuracy_-ii)*order_accuracy_+jj];
01183 phi.data()[(order_accuracy_-ii)*order_accuracy_+jj] = swap;
01184 }
01185 }
01186
01187 #if MTK_VERBOSE_LEVEL > 4
01188 std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01189 std::cout << phi << std::endl;
01190 #endif
01191
01192
01193
01194 mtk::Real *lamed{}; // Used to build big lambda.
01195
01196 try {
01197 lamed = new mtk::Real[dim_null_];
01198 } catch (std::bad_alloc &memory_allocation_exception) {
01199 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01200 std::endl;
01201 std::cerr << memory_allocation_exception.what() << std::endl;
01202 }
01203 memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01204
01205 for (auto ii = 0; ii < dim_null_; ++ii) {
01206 lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01207 }
01208
01209 #if MTK_VERBOSE_LEVEL > 3
01210 std::cout << "lamed =" << std::endl;
01211 for (auto ii = 0; ii < dim_null_; ++ii) {
01212 std::cout << std::setw(12) << lamed[ii] << std::endl;
01213 }
01214 std::cout << std::endl;
01215 #endif
01216
01217 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01218 mtk::Real temp = mtk::kZero;
01219 for(auto jj = 0; jj < dim_null_; ++jj) {
01220 temp = temp +
01221 lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01222 }
01223 hh[ii] = hh[ii] - temp;
01224 }
01225
01226 #if MTK_VERBOSE_LEVEL > 3
01227 std::cout << "big_lambda =" << std::endl;
01228 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01229 std::cout << std::setw(12) << hh[ii] << std::endl;
01230 }
01231 std::cout << std::endl;
01232 #endif
01233
01234 #ifdef MTK_VERBOSE_WEIGHTS
01235 int copy_result{1};
01236 #else
01237 int copy_result{};
01238 #endif
01239
01240 mtk::Real normerr_; // Norm of the error for the solution on each row.
01241
01242
01243
01244 int minrow_{std::numeric_limits<int>::infinity()};
01245
01246 mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_crs_,
01247 order_accuracy_)};
01248 mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01249
01250 #ifdef MTK_VERBOSE_WEIGHTS
01251 std::ofstream table("div_1d_" + std::to_string(order_accuracy_) +
01252 "_weights.tex");
01253 table << "\\begin{tabular}{c}{c}";
01254 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01255 table << 'c';
01256 }
01257 table << ":c}\\n\\toprule\\nRow & ";
01258 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01259 table << "$ q_{\text{" + std::to_string(ii) + "}}$ &";

```

```

01260 }
01261 table << " Relative error \\\n\\midrule\n";
01262 #endif
01263
01264 for(auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01265 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01266 order_accuracy_ + 1,
01267 order_accuracy_,
01268 order_accuracy_,
01269 hh,
01270 weights_cbs_,
01271 row_,
01272 mimetic_threshold_,
01273 copy_result);
01274 mtk::Real aux{normerr_/norm_};
01275
01276 #if MTK_VERBOSE_LEVEL > 2
01277 std::cout << "Relative norm: " << aux << " " << std::endl;
01278 std::cout << std::endl;
01279 #endif
01280
01281 if (aux < minnorm_) {
01282 minnorm_ = aux;
01283 minrow_ = row_;
01284 }
01285
01286 #ifdef MTK_VERBOSE_WEIGHTS
01287 table << std::to_string(row_ + 1) << " & ";
01288 if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01289 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01290 table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01291 }
01292 table << std::to_string(aux) << " \\\n" << std::endl;
01293 } else {
01294 table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
01295 "}{c}{\\emptyset$} & ";
01296 table << " - \\\n" << std::endl;
01297 }
01298 #endif
01299 }
01300
01301 #ifdef MTK_VERBOSE_WEIGHTS
01302 table << "\\midrule" << std::endl;
01303 table << "CRS weights:";
01304 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01305 table << " & " << std::to_string(weights_crs_[ii - 1]);
01306 }
01307 table << " & - \\\n\\bottomrule\n\\end{tabular}" << std::endl;
01308 table.close();
01309 #endif
01310
01311 #if MTK_VERBOSE_LEVEL > 3
01312 std::cout << "weights_CBSA + lambda (after brute force search):" <<
01313 std::endl;
01314 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01315 std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01316 }
01317 std::cout << std::endl;
01318 #endif
01319
01320 // After we know which row yields the smallest relative norm that row is
01321 // chosen to be the objective function and the result of the optimizer is
01322 // chosen to be the new weights_.
01323
01324 #if MTK_VERBOSE_LEVEL > 2
01325 std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01326 minrow_ + 1 << std::endl;
01327 std::cout << std::endl;
01328 #endif
01329
01330 copy_result = 1;
01331 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01332 order_accuracy_ + 1,
01333 order_accuracy_,
01334 order_accuracy_,
01335 hh,
01336 weights_cbs_,
01337 minrow_,

```

```

01340 mimetic_threshold_,
01341 copy_result);
01342 mtk::Real aux_{normerr_/norm_};
01343 #if MTK_VERBOSE_LEVEL > 2
01344 std::cout << "Relative norm: " << aux_ << std::endl;
01345 std::cout << std::endl;
01346 #endif
01347 delete [] lamed;
01348 lamed = nullptr;
01349 }
01350
01351
01352 delete [] hh;
01353 hh = nullptr;
01354
01355 return true;
01356 }
01357
01358 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01359
01360 #if MTK_VERBOSE_LEVEL > 3
01361 std::cout << "weights_CBSA + lambda =" << std::endl;
01362 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01363 std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01364 }
01365 std::cout << std::endl;
01366 #endif
01367
01368 mtk::Real *lambda{}; // Collection of bottom values from weights_.
01369
01370 try {
01371 lambda = new mtk::Real[dim_null_];
01372 } catch (std::bad_alloc &memory_allocation_exception) {
01373 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01374 std::endl;
01375 std::cerr << memory_allocation_exception.what() << std::endl;
01376 }
01377 memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01378
01379 for (auto ii = 0; ii < dim_null_; ++ii) {
01380 lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01381 }
01382
01383 #if MTK_VERBOSE_LEVEL > 3
01384 std::cout << "lambda =" << std::endl;
01385 for (auto ii = 0; ii < dim_null_; ++ii) {
01386 std::cout << std::setw(12) << lambda[ii] << std::endl;
01387 }
01388 std::cout << std::endl;
01389 #endif
01390
01391 mtk::Real *alpha{}; // Collection of alpha values.
01392
01393 try {
01394 alpha = new mtk::Real[dim_null_];
01395 } catch (std::bad_alloc &memory_allocation_exception) {
01396 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01397 std::endl;
01398 std::cerr << memory_allocation_exception.what() << std::endl;
01399 }
01400 memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01401
01402 for (auto ii = 0; ii < dim_null_; ++ii) {
01403 alpha[ii] = lambda[ii]/weights_cbs_[ii];
01404 }
01405
01406 #if MTK_VERBOSE_LEVEL > 3
01407 std::cout << "alpha =" << std::endl;
01408 for (auto ii = 0; ii < dim_null_; ++ii) {
01409 std::cout << std::setw(12) << alpha[ii] << std::endl;
01410 }
01411 std::cout << std::endl;
01412 #endif
01413
01414 try {
01415 mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01416 } catch (std::bad_alloc &memory_allocation_exception) {
01417 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```

```

01424 std::endl;
01425 std::cerr << memory_allocation_exception.what() << std::endl;
01426 }
01427 memset(mim_bndy_,
01428 mtk::kZero,
01429 sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01430
01431 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01432 for (auto jj = 0; jj < dim_null_; ++jj) {
01433 mim_bndy_[ii*dim_null_ + jj] =
01434 prem_apps_[ii*dim_null_ + jj] +
01435 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01436 }
01437 }
01438
01439 #if MTK_VERBOSE_LEVEL > 3
01440 std::cout << "Collection of mimetic approximations:" << std::endl;
01441 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01442 for (auto jj = 0; jj < dim_null_; ++jj) {
01443 std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01444 }
01445 std::cout << std::endl;
01446 }
01447 std::cout << std::endl;
01448 #endif
01449
01450 delete[] lambda;
01451 lambda = nullptr;
01452
01453 delete[] alpha;
01454 alpha = nullptr;
01455
01456 return true;
01457 }
01458
01459 bool mtk::Div1D::AssembleOperator(void) {
01460
01461 // The output array will have this form:
01462 // 1. The first entry of the array will contain used order order_accuracy_.
01463 // 2. The second entry of the array will contain the collection of
01464 // approximating coefficients for the interior of the grid.
01465 // 3. IF order_accuracy_ > 2, then the third entry will contain a collection
01466 // of weights.
01467 // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the
01468 // collections of approximating coefficients for the west boundary of the
01469 // grid.
01470
01471 if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01472 divergence_length_ =
01473 1 + order_accuracy_ + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01474 } else {
01475 divergence_length_ = 1 + order_accuracy_;
01476 }
01477
01478 #if MTK_VERBOSE_LEVEL > 2
01479 std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01480 #endif
01481
01482 try {
01483 divergence_ = new double[divergence_length_];
01484 } catch (std::bad_alloc &memory_allocation_exception) {
01485 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01486 std::endl;
01487 std::cerr << memory_allocation_exception.what() << std::endl;
01488 }
01489 memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01490
01491 divergence_[0] = order_accuracy_;
01492
01493 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01494 divergence_[ii + 1] = coeffs_interior_[ii];
01495 }
01496
01497 if (order_accuracy_ > 2) {
01498 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01499 divergence_[1 + order_accuracy_ + ii] = weights_cbs_[ii];
01500 }
01501 }
01502 }

```



```

01508
01511
01512 if (order_accuracy_ > 2) {
01513 auto offset = (2*order_accuracy_ + 1);
01514 int mm{};
01515 for (auto ii = 0; ii < dim_null_; ++ii) {
01516 for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01517 divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01518 ++mm;
01519 }
01520 }
01521 }
01522
01523 #if MTK_VERBOSE_LEVEL > 1
01524 std::cout << "1D " << order_accuracy_ << "--order div built!" << std::endl;
01525 std::cout << std::endl;
01526 #endif
01527
01528 return true;
01529 }

```

## 18.85 src/mtk\_div\_2d.cc File Reference

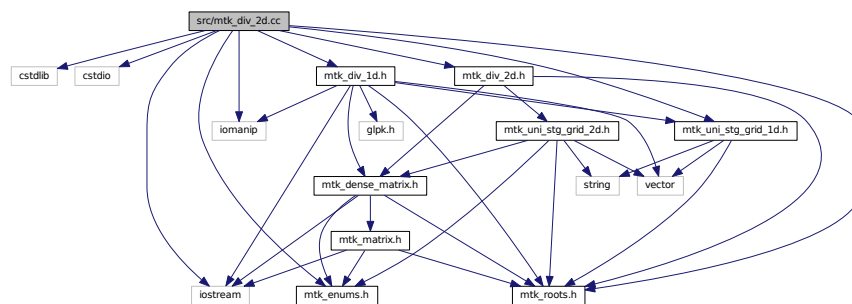
Implements the class Div2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"

```

Include dependency graph for mtk\_div\_2d.cc:



### 18.85.1 Detailed Description

This class implements a 2D divergence matrix operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_2d.cc](#).

## 18.86 mtk\_div\_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_enums.h"
00065 #include "mtk_uni_stg_grid_ld.h"
00066 #include "mtk_div_ld.h"
00067 #include "mtk_div_2d.h"
00068
00069 mtk::Div2D::Div2D():
00070 order_accuracy_(),
00071 mimetic_threshold_() {}
00072
00073 mtk::Div2D::Div2D(const Div2D &div):
00074 order_accuracy_(div.order_accuracy_),
00075 mimetic_threshold_(div.mimetic_threshold_) {}
00076
00077 mtk::Div2D::~Div2D() {}
00078
00079 bool mtk::Div2D::ConstructDiv2D(const
 mtk::UniStgGrid2D &grid,
 int order_accuracy,
 mtk::Real mimetic_threshold) {
00080
00081
00082
00083 int num_cells_x = grid.num_cells_x();
00084 int num_cells_y = grid.num_cells_y();
00085
00086 int mx = num_cells_x + 2; // Dx vertical dimension.

```

```

00087 int nx = num_cells_x + 1; // Dx horizontal dimension.
00088 int my = num_cells_y + 2; // Dy vertical dimension.
00089 int ny = num_cells_y + 1; // Dy horizontal dimension.
00090
00091 mtk::Div1D div;
00092
00093 bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00094
00095 #ifdef MTK_PERFORM_PREVENTIONS
00096 if (!info) {
00097 std::cerr << "Mimetic div could not be built." << std::endl;
00098 return info;
00099 }
00100 #endif
00101
00102 auto west = grid.west_bndy();
00103 auto east = grid.east_bndy();
00104 auto south = grid.south_bndy();
00105 auto north = grid.east_bndy();
00106
00107 mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00108 mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00109
00110 mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00111 mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00112
00113 bool padded{true};
00114 bool transpose{false};
00115
00116 mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00117 mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00118
00119 mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00120 mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00121
00122 #if MTK_VERBOSE_LEVEL > 2
00123 std::cout << "Dx: " << mx << " by " << nx << std::endl;
00124 std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;
00125 std::cout << "Dy: " << my << " by " << ny << std::endl;
00126 std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00127 std::cout << "Div 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00128 nx*ny << std::endl;
00129 #endif
00130
00131 mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00132
00133 for (auto ii = 0; ii < mx*my; ii++) {
00134 for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00135 d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00136 }
00137 for (auto kk = 0; kk < ny*num_cells_x; kk++) {
00138 d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00139 }
00140 }
00141
00142 divergence_ = d2d;
00143
00144 return info;
00145 }
00146
00147 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() const {
00148 return divergence_;
00149 }
00150 }

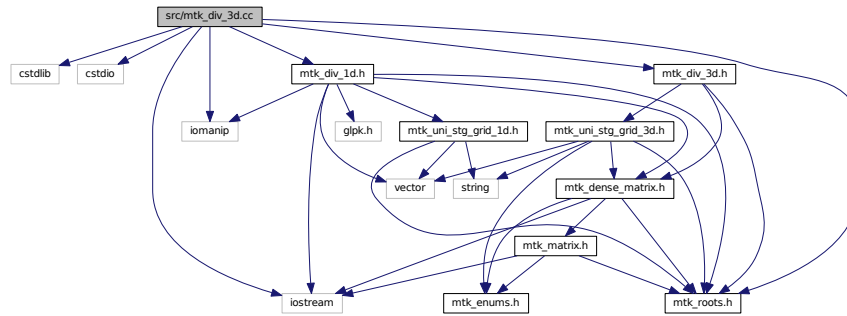
```

## 18.87 src/mtk\_div\_3d.cc File Reference

Implements the class Div3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_div_1d.h"
#include "mtk_div_3d.h"
```

Include dependency graph for mtk\_div\_3d.cc:



### 18.87.1 Detailed Description

This class implements a 3D divergence operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_3d.cc](#).

## 18.88 mtk\_div\_3d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
```

```

00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_div_ld.h"
00065 #include "mtk_div_3d.h"
00066
00067 mtk::Div3D::Div3D():
00068 order_accuracy_(),
00069 mimetic_threshold_() {}
00070
00071 mtk::Div3D::Div3D(const Div3D &grad):
00072 order_accuracy_(grad.order_accuracy_),
00073 mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Div3D::~Div3D() {}
00076
00077 bool mtk::Div3D::ConstructDiv3D(const
 mtk::UniStgGrid3D &grid,
 int order_accuracy,
 mtk::Real mimetic_threshold) {
00078
00079
00080
00081 int num_cells_x = grid.num_cells_x();
00082 int num_cells_y = grid.num_cells_y();
00083 int num_cells_z = grid.num_cells_z();
00084
00085 int mx = num_cells_x + 1; // Dx vertical dimension.
00086 int nx = num_cells_x + 2; // Dx horizontal dimension.
00087 int my = num_cells_y + 1; // Dy vertical dimension.
00088 int ny = num_cells_y + 2; // Dy horizontal dimension.
00089 int mz = num_cells_z + 1; // Dz vertical dimension.
00090 int nz = num_cells_z + 2; // Dz horizontal dimension.
00091
00092 mtk::Div1D div;
00093
00094 bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00095
00096 #ifdef MTK_PERFORM_PREVENTIONS
00097 if (!info) {
00098 std::cerr << "Mimetic div could not be built." << std::endl;
00099 return info;
00100 }
00101 #endif
00102
00103 auto west = grid.west_bndy();
00104 auto east = grid.east_bndy();
00105 auto south = grid.south_bndy();
00106 auto north = grid.east_bndy();
00107 auto bottom = grid.bottom_bndy();
00108 auto top = grid.top_bndy();
00109
00110 mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00111 mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00112 mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z);

```

```

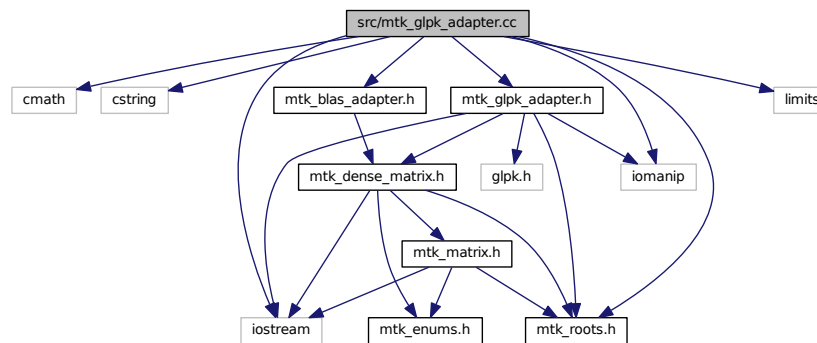
00113
00114 mtk::DenseMatrix Dx(div.ReturnAsDenseMatrix(grid_x));
00115 mtk::DenseMatrix Dy(div.ReturnAsDenseMatrix(grid_y));
00116 mtk::DenseMatrix Dz(div.ReturnAsDenseMatrix(grid_z));
00117
00118 bool padded{true};
00119 bool transpose{false};
00120
00121 mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00122 mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00123 mtk::DenseMatrix iz(num_cells_z, padded, transpose);
00124
00126
00127 mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(iz, iy));
00128 mtk::DenseMatrix dx(mtk::DenseMatrix::Kron(aux1, Dx));
00129
00131
00132 mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(iz, Dy));
00133 mtk::DenseMatrix dy(mtk::DenseMatrix::Kron(aux2, ix));
00134
00136
00137 mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Dz, iy));
00138 mtk::DenseMatrix dz(mtk::DenseMatrix::Kron(aux3, ix));
00139
00140 #if MTK_VERBOSE_LEVEL > 2
00141 std::cout << "Dx: " << mx << " by " << nx << std::endl;
00142 std::cout << "Ix: " << num_cells_x << " by " << nx << std::endl;
00143 std::cout << "Dy: " << my << " by " << ny << std::endl;
00144 std::cout << "Iy: " << num_cells_y << " by " << ny << std::endl;
00145 std::cout << "Dz: " << mz << " by " << nz << std::endl;
00146 std::cout << "Iz: " << num_cells_z << " by " << nz << std::endl;
00147 #endif
00148
00150
00151 int total_rows{nx*ny*nz};
00152 int total_cols{mx*num_cells_y*num_cells_z +
00153 num_cells_x*my*num_cells_z +
00154 num_cells_x*num_cells_y*mz};
00155
00156 #if MTK_VERBOSE_LEVEL > 2
00157 std::cout << "Div 3D: " << total_rows << " by " << total_cols << std::endl;
00158 #endif
00159
00160 mtk::DenseMatrix d3d(total_rows, total_cols);
00161
00162 for (auto ii = 0; ii < total_rows; ++ii) {
00163
00164 for (auto jj = 0; jj < mx*num_cells_y*num_cells_z; ++jj) {
00165 d3d.SetValue(ii, jj, dx.GetValue(ii, jj));
00166 }
00167
00168 int offset = mx*num_cells_y*num_cells_z;
00169
00170 for(auto kk = 0; kk < num_cells_x*my*num_cells_z; ++kk) {
00171 d3d.SetValue(ii, kk + offset, dy.GetValue(ii, kk));
00172 }
00173
00174 offset += num_cells_x*my*num_cells_z;
00175
00176 for(auto ll = 0; ll < num_cells_x*num_cells_y*mz; ++ll) {
00177 d3d.SetValue(ii, ll + offset, dz.GetValue(ii, ll));
00178 }
00179 }
00180
00181 divergence_ = d3d;
00182
00183 return info;
00184 }
00185
00186 mtk::DenseMatrix mtk::Div3D::ReturnAsDenseMatrix() const {
00187
00188 return divergence_;
00189 }

```

## 18.89 src/mtk\_glpk\_adapter.cc File Reference

Adapter class for the GLPK API.

```
#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_glpk_adapter.h"
Include dependency graph for mtk_glpk_adapter.cc:
```



### 18.89.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the GLPK.

The **GLPK (GNU Linear Programming Kit)** package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

See also

<http://www.gnu.org/software/glpk/>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_glpk\\_adapter.cc](#).

## 18.90 mtk\_glpk\_adapter.cc

00001

```

00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #include <cmath>
00067 #include <cstring>
00068
00069 #include <iostream>
00070 #include <iomanip>
00071 #include <limits>
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_blas_adapter.h"
00075 #include "mtk_glpk_adapter.h"
00076
00077 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
00078 mtk::Real *A,
00079 int nrows,
00080 int ncols,
00081 int kk,
00082 mtk::Real *hh,
00083 mtk::Real *qq,
00084 int robjective,
00085 mtk::Real mimetic_threshold,
00086 int copy) {
00087 #if MTK_DEBUG_LEVEL > 0
00088 char mps_file_name[18]; // File name for the MPS files.
00089 #endif
00090 char rname[5]; // Row name.
00091 char cname[5]; // Column name.
00092
00093 glp_prob *lp; // Linear programming problem.
00094
00095 int *ia; // Array for the problem.
00096 int *ja; // Array for the problem.
00097
00098 int problem_size; // Size of the problem.
00099 int lp_nrows; // Number of rows.

```



```

00100 int lp_ncols; // Number of columns.
00101 int matsize; // Size of the matrix.
00102 int glp_index{1}; // Index of the objective function.
00103 int ii; // Iterator.
00104 int jj; // Iterator.
00105
00106 mtk::Real *ar; // Array for the problem.
00107 mtk::Real *objective; // Array containing the objective function.
00108 mtk::Real *rhs; // Array containing the rhs.
00109 mtk::Real *err; // Array of errors.
00110
00111 mtk::Real x1; // Norm-2 of the error.
00112
00113 #if MTK_DEBUG_LEVEL > 0
00114 mtk::Real obj_value; // Value of the objective function.
00115 #endif
00116
00117 lp_nrows = kk;
00118 lp_ncols = kk;
00119
00120 matsize = lp_nrows*lp_ncols;
00121
00122 problem_size = lp_nrows*lp_ncols + 1;
00123
00124 try {
00125 ia = new int[problem_size];
00126 } catch (std::bad_alloc &memory_allocation_exception) {
00127 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00128 std::endl;
00129 std::cerr << memory_allocation_exception.what() << std::endl;
00130 }
00131 memset(ia, 0, sizeof(ia[0])*problem_size);
00132
00133 try {
00134 ja = new int[problem_size];
00135 } catch (std::bad_alloc &memory_allocation_exception) {
00136 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00137 std::endl;
00138 std::cerr << memory_allocation_exception.what() << std::endl;
00139 }
00140 memset(ja, 0, sizeof(ja[0])*problem_size);
00141
00142 try {
00143 ar = new mtk::Real[problem_size];
00144 } catch (std::bad_alloc &memory_allocation_exception) {
00145 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00146 std::endl;
00147 std::cerr << memory_allocation_exception.what() << std::endl;
00148 }
00149 memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00150
00151 try {
00152 objective = new mtk::Real[lp_ncols + 1];
00153 } catch (std::bad_alloc &memory_allocation_exception) {
00154 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00155 std::endl;
00156 std::cerr << memory_allocation_exception.what() << std::endl;
00157 }
00158 memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00159
00160 try {
00161 rhs = new mtk::Real[lp_nrows + 1];
00162 } catch (std::bad_alloc &memory_allocation_exception) {
00163 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00164 std::endl;
00165 std::cerr << memory_allocation_exception.what() << std::endl;
00166 }
00167 memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00168
00169 try {
00170 err = new mtk::Real[lp_nrows];
00171 } catch (std::bad_alloc &memory_allocation_exception) {
00172 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00173 std::endl;
00174 std::cerr << memory_allocation_exception.what() << std::endl;
00175 }
00176 memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00177
00178 #if MTK_DEBUG_LEVEL > 0
00179 std::cout << "Problem size: " << problem_size << std::endl;
00180

```

```

00183 std::cout << "lp_nrows = " << lp_nrows << std::endl;
00184 std::cout << "lp_ncols = " << lp_ncols << std::endl;
00185 std::cout << std::endl;
00186 #endif
00187
00188 lp = glp_create_prob();
00189
00190 glp_set_prob_name (lp, "mtk::GLPKAdapter::Simplex");
00191
00192 glp_set_obj_dir (lp, GLP_MIN);
00193
00194
00195
00196 glp_add_rows(lp, lp_nrows);
00197
00198 for (ii = 1; ii <= lp_nrows; ++ii) {
00199 sprintf(rname, "R%02d",ii);
00200 glp_set_row_name(lp, ii, rname);
00201 }
00202
00203 glp_add_cols(lp, lp_ncols);
00204
00205 for (ii = 1; ii <= lp_ncols; ++ii) {
00206 sprintf(cname, "Q%02d",ii);
00207 glp_set_col_name (lp, ii, cname);
00208 }
00209
00210
00211
00212 #if MTK_DEBUG_LEVEL>0
00213 std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
00214 #endif
00215 for (jj = 0; jj < kk; ++jj) {
00216 objective[glp_index] = A[jj + robjective * ncols];
00217 glp_index++;
00218 }
00219 #if MTK_DEBUG_LEVEL >0
00220 std::cout << std::endl;
00221 #endif
00222
00223
00224
00225 glp_index = 1;
00226 rhs[0] = mtk::kZero;
00227 for (ii = 0; ii <= lp_nrows; ++ii) {
00228 if (ii != robjective) {
00229 rhs[glp_index] = hh[ii];
00230 glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00231 glp_index++;
00232 }
00233 }
00234
00235 #if MTK_DEBUG_LEVEL > 0
00236 std::cout << "rhs =" << std::endl;
00237 for (auto ii = 0; ii < lp_nrows; ++ii) {
00238 std::cout << std::setw(15) << rhs[ii] << std::endl;
00239 }
00240 std::cout << std::endl;
00241 #endif
00242
00243
00244
00245 for (ii = 1; ii <= lp_ncols; ++ii) {
00246 glp_set_obj_coef (lp, ii, objective[ii]);
00247 }
00248
00249
00250
00251 for (ii = 1; ii <= lp_ncols; ++ii) {
00252 glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00253 }
00254
00255
00256 glp_index = 1;
00257 for (ii = 0; ii <= kk; ++ii) {
00258 for (jj = 0; jj < kk; ++jj) {
00259 if (ii != robjective) {
00260 ar[glp_index] = A[jj + ii * ncols];
00261 glp_index++;
00262 }
00263 }
00264 }
00265
00266
00267 glp_index = 0;
00268
00269 for (ii = 1; ii < problem_size; ++ii) {

```

```

00270 if ((ii - 1) % lp_ncols) == 0) {
00271 glp_index++;
00272 }
00273 ia[ii] = glp_index;
00274 ja[ii] = (ii - 1) % lp_ncols + 1;
00275 }
00276
00277 glp_load_matrix (lp, matsize, ia, ja, ar);
00278
00279 #if MTK_DEBUG_LEVEL > 0
00280 sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00281 glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00282 #endif
00283
00285 glp_simplex (lp, nullptr);
00286
00287 // Check status of the solution.
00288
00289 if (glp_get_status(lp) == GLP_OPT) {
00290
00291 for(ii = 1; ii <= lp_ncols; ++ii) {
00292 err[ii - 1] = qq[ii - 1] - glp_get_col_prim(lp, ii);
00293 }
00294
00295 #if MTK_DEBUG_LEVEL > 0
00296 obj_value = glp_get_obj_val (lp);
00297 std::cout << std::setw(12) << "CBS" << std::setw(12) << "CRS" << std::endl;
00298 for (ii = 0; ii < lp_ncols; ++ii) {
00299 std::cout << "q_" << ii + 1 << " = " << std::setw(12) <<
00300 glp_get_col_prim(lp, ii + 1) << std::setw(12) << qq[ii] << std::endl;
00301 }
00302 std::cout << "Objective function value (row " << robjective + 1 << ") = " <<
00303 obj_value << std::endl;
00304 #endif
00305
00306 if (copy) {
00307 for(ii = 0; ii < lp_ncols; ++ii) {
00308 qq[ii] = glp_get_col_prim(lp, ii + 1);
00309 }
00310 // Preserve the bottom values of qq.
00311 }
00312
00313 x1 = mtk::BLASAdapter::RealNRM2(err, lp_ncols);
00314
00315 } else {
00316 x1 = std::numeric_limits<mtk::Real>::infinity();
00317 }
00318
00319 glp_delete_prob (lp);
00320 glp_free_env ();
00321
00322 delete [] ia;
00323 delete [] ja;
00324 delete [] ar;
00325 delete [] objective;
00326 delete [] rhs;
00327 delete [] err;
00328
00329 return x1;
00330 }
00331 }

```

## 18.91 src/mtk\_grad\_1d.cc File Reference

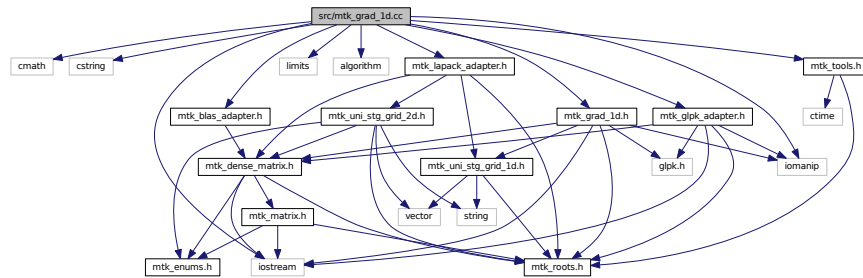
Implements the class Grad1D.

```

#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_grad_1d.h"

```

Include dependency graph for mtk\_grad\_1d.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::Grad1D &in)`

### 18.91.1 Detailed Description

This class implements a 1D gradient matrix operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Overload ostream operator as in [mtk::Lap1D](#).

**Todo** Implement creation of ■ w. [mtk::BLASAdapter](#).

Definition in file [mtk\\_grad\\_1d.cc](#).

## 18.92 mtk\_grad\_1d.cc

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075
00076 #include "mtk_blas_adapter.h"
00077 #include "mtk_lapack_adapter.h"
00078 #include "mtk_glpk_adapter.h"
00079
00080 #include "mtk_grad_1d.h"
00081
00082 namespace mtk {
00083
00084 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &in) {
00085
00086 stream << "gradient_[0] = " << std::setw(9) << in.gradient_[0] << std::endl;
00087
00088 stream << "gradient_[1:" << in.order_accuracy_ << "] = ";
00089 for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {

```

```

00094 stream << std::setw(9) << in.gradient_[ii] << " ";
00095 }
00096 stream << std::endl;
00097
00098
00099
00100 stream << "gradient[" << in.order_accuracy_ + 1 << ":" <<
00101 2*in.order_accuracy_ << "]" = ";
00102 for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
order_accuracy_; ++ii) {
00103 stream << std::setw(9) << in.gradient_[ii] << " ";
00104 }
00105 stream << std::endl;
00106
00107
00108
00109 int offset{2*in.order_accuracy_ + 1};
00110 int mm {};
00111
00112 stream << "gradient[" << offset + mm << ":" <<
00113 offset + mm + in.num_bndy_coeffs_ - 1 << "]" = ";
00114
00115 if (in.order_accuracy_ > mtk::kDefaultOrderAccuracy) {
00116 for (auto ii = 0; ii < in.num_bndy_approxs_ ; ++ii) {
00117 for (auto jj = 0; jj < in.num_bndy_coeffs_ ; jj++) {
00118 auto value = in.gradient_[offset + (mm)];
00119 stream << std::setw(9) << value << " ";
00120 mm++;
00121 }
00122 }
00123 } else {
00124 stream << std::setw(9) << in.gradient_[offset + 0] << ' ';
00125 stream << std::setw(9) << in.gradient_[offset + 1] << ' ';
00126 stream << std::setw(9) << in.gradient_[offset + 2] << ' ';
00127 }
00128 stream << std::endl;
00129
00130 return stream;
00131 }
00132 }
00133
00134 mtk::Grad1D::Grad1D():
00135 order_accuracy_(mtk::kDefaultOrderAccuracy),
00136 dim_null_(),
00137 num_bndy_approxs_(),
00138 num_bndy_coeffs_(),
00139 gradient_length_(),
00140 minrow_(),
00141 row_(),
00142 coeffs_interior_(),
00143 prem_apps_(),
00144 weights_crs_(),
00145 weights_cbs_(),
00146 mim_bndy_(),
00147 gradient_(),
00148 mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00149
00150 mtk::Grad1D::Grad1D(const Grad1D &grad):
00151 order_accuracy_(grad.order_accuracy_),
00152 dim_null_(grad.dim_null_),
00153 num_bndy_approxs_(grad.num_bndy_approxs_),
00154 num_bndy_coeffs_(grad.num_bndy_coeffs_),
00155 gradient_length_(grad.gradient_length_),
00156 minrow_(grad.minrow_),
00157 row_(grad.row_),
00158 coeffs_interior_(grad.coeffs_interior_),
00159 prem_apps_(grad.prem_apps_),
00160 weights_crs_(grad.weights_crs_),
00161 weights_cbs_(grad.weights_cbs_),
00162 mim_bndy_(grad.mim_bndy_),
00163 gradient_(grad.gradient_),
00164 mimetic_threshold_(grad.mimetic_threshold_) {}
00165
00166 mtk::Grad1D::~Grad1D() {
00167
00168 delete[] coeffs_interior_;
00169 coeffs_interior_ = nullptr;
00170
00171 delete[] prem_apps_;
00172 prem_apps_ = nullptr;
00173
00174 delete[] weights_crs_;
00175 weights_crs_ = nullptr;

```

```

00176
00177 delete[] weights_cbs_;
00178 weights_cbs_ = nullptr;
00179
00180 delete[] mim_bndy_;
00181 mim_bndy_ = nullptr;
00182
00183 delete[] gradient_;
00184 gradient_ = nullptr;
00185 }
00186
00187 bool mtk::Grad1D::ConstructGrad1D(int order_accuracy,
00188 Real mimetic_threshold) {
00189
00190 #ifdef MTK_PERFORM_PREVENTIONS
00191 mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00192 mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00193 mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00194 __FILE__, __LINE__, __func__);
00195
00196 if (order_accuracy >= mtk::kCriticalOrderAccuracyGrad) {
00197 std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00198 }
00199
00200 std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00201 std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00202 #endif
00203
00204 order_accuracy_ = order_accuracy;
00205 mimetic_threshold_ = mimetic_threshold;
00206
00207 bool abort_construction = ComputeStencilInteriorGrid();
00208
00209 #ifdef MTK_PERFORM_PREVENTIONS
00210 if (!abort_construction) {
00211 std::cerr << "Could NOT complete stage 1." << std::endl;
00212 std::cerr << "Exiting..." << std::endl;
00213 return false;
00214 }
00215 #endif
00216
00217 // At this point, we already have the values for the interior stencil stored
00218 // in the coeffs_interior_ array.
00219
00220 dim_null_ = order_accuracy/2 - 1;
00221 num_bndy_approxs_ = dim_null_ + 1;
00222
00223 #ifdef MTK_PRECISION_DOUBLE
00224 num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy)/2.0);
00225 #else
00226 num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy)/2.0f);
00227 #endif
00228
00229
00230 // For this we will follow recommendations given in:
00231 //
00232 // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00233 //
00234 // We will compute the QR Factorization of the transpose, as in the
00235 // following (MATLAB) pseudo-code:
00236 //
00237 // [Q,R] = qr(V'); % Full QR as defined in
00238 // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00239 //
00240 // null-space = Q(:, last (order_accuracy/2 - 1) columns of Q);
00241 //
00242 // However, given the nature of the Vandermonde matrices we've just
00243 // computed, they all possess the same null-space. Therefore, we impose the
00244 // convention of computing the null-space of the first Vandermonde matrix
00245 // (west boundary).
00246 //
00247 // In the case of the gradient, the first Vandermonde system has a unique
00248 // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00249 // matrix used to assemble said system, will have an empty null-space.
00250 //
00251 // Therefore, we only compute a rational basis for the case of order higher
00252 // than second.
00253
00254 if (dim_null_ > 0) {
00255
00256
00257

```

```

00258 abort_construction = ComputeRationalBasisNullSpace();
00259
00260 #ifdef MTK_PERFORM_PREVENTIONS
00261 if (!abort_construction) {
00262 std::cerr << "Could NOT complete stage 2.1." << std::endl;
00263 std::cerr << "Exiting..." << std::endl;
00264 return false;
00265 }
00266 #endif
00267 }
00268
00270 abort_construction = ComputePreliminaryApproximations();
00271
00272 #ifdef MTK_PERFORM_PREVENTIONS
00273 if (!abort_construction) {
00274 std::cerr << "Could NOT complete stage 2.2." << std::endl;
00275 std::cerr << "Exiting..." << std::endl;
00276 return false;
00277 }
00278 #endif
00279
00281 abort_construction = ComputeWeights();
00282
00283 #ifdef MTK_PERFORM_PREVENTIONS
00284 if (!abort_construction) {
00285 std::cerr << "Could NOT complete stage 2.3." << std::endl;
00286 std::cerr << "Exiting..." << std::endl;
00287 return false;
00288 }
00289 #endif
00290
00292 if (dim_null_ > 0) {
00293
00294 abort_construction = ComputeStencilBoundaryGrid();
00295
00296 #ifdef MTK_PERFORM_PREVENTIONS
00297 if (!abort_construction) {
00298 std::cerr << "Could NOT complete stage 2.4." << std::endl;
00299 std::cerr << "Exiting..." << std::endl;
00300 return false;
00301 }
00302 #endif
00303 }
00304
00306
00307 // Once we have the following three collections of data:
00308 // (a) the coefficients for the interior,
00309 // (b) the coefficients for the boundary (if it applies),
00310 // (c) and the weights (if it applies),
00311 // we will store everything in the output array:
00312
00313 abort_construction = AssembleOperator();
00314
00315 #ifdef MTK_PERFORM_PREVENTIONS
00316 if (!abort_construction) {
00317 std::cerr << "Could NOT complete stage 3." << std::endl;
00318 std::cerr << "Exiting..." << std::endl;
00319 return false;
00320 }
00321 #endif
00322
00323 return true;
00324 }
00325
00326 int mtk::Grad1D::num_bndy_coeffs() const {
00327
00328 return num_bndy_coeffs_;
00329 }
00330
00331 mtk::Real *mtk::Grad1D::coeffs_interior() const {
00332
00333 return coeffs_interior_;
00334 }
00335
00336 mtk::Real *mtk::Grad1D::weights_crs() const {
00337
00338 return weights_crs_;
00339 }
00340
00341 mtk::Real *mtk::Grad1D::weights_cbs() const {
00342

```



```

00343 return weights_cbs_;
00344 }
00345
00346 mtk::DenseMatrix mtk::Grad1D::mim_bndy() const {
00347
00348 mtk::DenseMatrix xx(dim_null_ + 1, 3*order_accuracy_/2);
00349
00350 auto counter = 0;
00351 for (auto ii = 0; ii < dim_null_ + 1; ++ii) {
00352 for(auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00353 xx.SetValue(ii,jj, gradient_[2*order_accuracy_ + 1 + counter]);
00354 counter++;
00355 }
00356 }
00357
00358 return xx;
00359 }
00360
00361 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
 mtk::Real west,
00362 mtk::Real east,
00363 int num_cells_x) const {
00364
00365 int nn{num_cells_x}; // Number of cells on the grid.
00366
00367 #ifdef MTK_PERFORM_PREVENTIONS
00368 mtk::Tools::Prevent(east < west, __FILE__, __LINE__, __func__);
00369 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00370 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00371 #endif
00372
00373 mtk::Real delta_x = (east - west)/((mtk::Real) num_cells_x);
00374
00375 mtk::Real inv_delta_x{mtk::kOne/delta_x};
00376
00377 int gg_num_rows = nn + 1;
00378 int gg_num_cols = nn + 2;
00379 int elements_per_row = num_bndy_coeffs_;
00380 int num_extra_rows = order_accuracy_/2;
00381
00382 // Output matrix featuring sizes for gradient operators.
00383 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00384
00385 auto ee_index = 0;
00386 for (auto ii = 0; ii < num_extra_rows; ii++) {
00387 auto cc = 0;
00388 for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00389 if(cc >= elements_per_row) {
00390 out.SetValue(ii, jj, mtk::kZero);
00391 } else {
00392 out.SetValue(ii, jj,
00393 gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00394 cc++;
00395 }
00396 }
00397 }
00398
00399 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00400 auto jj = ii - num_extra_rows + 1;
00401 for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00402 out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00403 }
00404 }
00405
00406 ee_index = 0;
00407 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00408 auto cc = 0;
00409 for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00410 if(cc >= elements_per_row) {
00411 out.SetValue(ii, jj, mtk::kZero);
00412 } else {
00413 out.SetValue(ii, jj,
00414 -gradient_[2*order_accuracy_ + 1 +
00415 ee_index++]*inv_delta_x);
00416 cc++;
00417 }
00418 }
00419 }
00420
00421 }
00422
00423 }
00424
00425 }

```

```

00426
00427 return out;
00428 }
00429
00430 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00431 const UniStgGrid1D &grid) const {
00432
00433 int nn{grid.num_cells_x()}; // Number of cells on the grid.
00434
00435 #ifdef MTK_PERFORM_PREVENTIONS
00436 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00437 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00438 #endif
00439
00440 mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00441
00442 int gg_num_rows = nn + 1;
00443 int gg_num_cols = nn + 2;
00444 int elements_per_row = num_bndy_coeffs_;
00445 int num_extra_rows = order_accuracy_/2;
00446
00447 // Output matrix featuring sizes for gradient operators.
00448 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00449
00450
00451 auto ee_index = 0;
00452 for (auto ii = 0; ii < num_extra_rows; ii++) {
00453 auto cc = 0;
00454 for (auto jj = 0; jj < gg_num_cols; jj++) {
00455 if (cc >= elements_per_row) {
00456 out.SetValue(ii, jj, mtk::kZero);
00457 } else {
00458 out.SetValue(ii, jj,
00459 gradient_[2*order_accuracy_ + 1 + ee_index++] * inv_delta_x);
00460 cc++;
00461 }
00462 }
00463 }
00464 }
00465
00466
00467 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00468 auto jj = ii - num_extra_rows + 1;
00469 for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00470 out.SetValue(ii, jj, coeffs_interior_[cc] * inv_delta_x);
00471 }
00472 }
00473 }
00474
00475
00476 ee_index = 0;
00477 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00478 auto cc = 0;
00479 for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00480 if (cc >= elements_per_row) {
00481 out.SetValue(ii, jj, mtk::kZero);
00482 } else {
00483 out.SetValue(ii, jj,
00484 -gradient_[2*order_accuracy_ + 1 + ee_index++] * inv_delta_x);
00485 cc++;
00486 }
00487 }
00488 }
00489 }
00490
00491 return out;
00492 }
00493
00494 mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix
00495 (
00496 int num_cells_x) const {
00497
00498 int nn{num_cells_x}; // Number of cells on the grid.
00499
00500 #ifdef MTK_PERFORM_PREVENTIONS
00501 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00502 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00503 #endif
00504
00505 int gg_num_rows = nn + 1;
00506 int gg_num_cols = nn + 2;
00507 int elements_per_row = num_bndy_coeffs_;
00508 int num_extra_rows = order_accuracy_/2;

```

```

00509 // Output matrix featuring sizes for gradient operators.
00510 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00511
00513
00514 auto ee_index = 0;
00515 for (auto ii = 0; ii < num_extra_rows; ii++) {
00516 auto cc = 0;
00517 for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00518 if(cc >= elements_per_row) {
00519 out.SetValue(ii, jj, mtk::kZero);
00520 } else {
00521 out.SetValue(ii, jj,
00522 gradient_[2*order_accuracy_ + 1 + ee_index++]);
00523 cc++;
00524 }
00525 }
00526 }
00527
00529
00530 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00531 auto jj = ii - num_extra_rows + 1;
00532 for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00533 out.SetValue(ii, jj, coeffs_interior_[cc]);
00534 }
00535 }
00536
00538
00539 ee_index = 0;
00540 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00541 auto cc = 0;
00542 for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00543 if(cc >= elements_per_row) {
00544 out.SetValue(ii, jj, mtk::kZero);
00545 } else {
00546 out.SetValue(ii, jj,
00547 -gradient_[2*order_accuracy_ + 1 + ee_index++]);
00548 cc++;
00549 }
00550 }
00551 }
00552
00553 return out;
00554 }
00555
00556 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00557
00559
00560 mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00561
00562 try {
00563 pp = new mtk::Real[order_accuracy_];
00564 } catch (std::bad_alloc &memory_allocation_exception) {
00565 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00566 std::endl;
00567 std::cerr << memory_allocation_exception.what() << std::endl;
00568 }
00569 memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00570
00571 #ifdef MTK_PRECISION_DOUBLE
00572 pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00573 #else
00574 pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00575 #endif
00576
00577 for (auto ii = 1; ii < order_accuracy_; ++ii) {
00578 pp[ii] = pp[ii - 1] + mtk::kOne;
00579 }
00580
00581 #if MTK_VERBOSE_LEVEL > 3
00582 std::cout << "pp =" << std::endl;
00583 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00584 std::cout << std::setw(12) << pp[ii];
00585 }
00586 std::cout << std::endl << std::endl;
00587 #endif
00588
00590
00591 bool transpose{false};
00592
00593 mtk::DenseMatrix vander_matrix(pp, order_accuracy_, order_accuracy_, transpose);
00594

```

```

00595 #if MTK_VERBOSE_LEVEL > 4
00596 std::cout << "vander_matrix = " << std::endl;
00597 std::cout << vander_matrix << std::endl << std::endl;
00598 #endif
00599
00601
00602 try {
00603 coeffs_interior_ = new mtk::Real[order_accuracy_];
00604 } catch (std::bad_alloc &memory_allocation_exception) {
00605 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00606 std::endl;
00607 std::cerr << memory_allocation_exception.what() << std::endl;
00608 }
00609 memset(coeffs_interior_, mtk::kZero,
00610 sizeof(coeffs_interior_[0])*order_accuracy_);
00611
00612 coeffs_interior_[1] = mtk::kOne;
00613
00614 #if MTK_VERBOSE_LEVEL > 3
00615 std::cout << "oo =" << std::endl;
00616 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00617 std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00618 }
00619 std::cout << std::endl;
00620 #endif
00621
00622
00623
00624 int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00625 coeffs_interior_)};
00626
00627 #ifdef MTK_PERFORM_PREVENTIONS
00628 if (!info) {
00629 std::cout << "System solved! Interior stencil attained!" << std::endl;
00630 std::cout << std::endl;
00631 }
00632 else {
00633 std::cerr << "Something wrong solving system! info = " << info << std::endl;
00634 std::cerr << "Exiting..." << std::endl;
00635 return false;
00636 }
00637 #endif
00638
00639 #if MTK_VERBOSE_LEVEL > 3
00640 std::cout << "coeffs_interior_ =" << std::endl;
00641 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00642 std::cout << std::setw(12) << coeffs_interior_[ii];
00643 }
00644 std::cout << std::endl << std::endl;
00645 #endif
00646
00647 delete [] pp;
00648 pp = nullptr;
00649
00650 return true;
00651 }
00652
00653 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00654
00655
00656
00657 mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00658
00659 try {
00660 gg = new mtk::Real[num_bndy_coeffs_];
00661 } catch (std::bad_alloc &memory_allocation_exception) {
00662 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00663 std::endl;
00664 std::cerr << memory_allocation_exception.what() << std::endl;
00665 }
00666 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00667
00668 #ifdef MTK_PRECISION_DOUBLE
00669 gg[1] = 1.0/2.0;
00670 #else
00671 gg[1] = 1.0f/2.0f;
00672 #endif
00673 for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00674 gg[ii] = gg[ii - 1] + mtk::kOne;
00675 }
00676
00677 #if MTK_VERBOSE_LEVEL > 3
00678 std::cout << "gg =" << std::endl;

```

```

00679 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00680 std::cout << std::setw(12) << gg[ii];
00681 }
00682 std::cout << std::endl << std::endl;
00683 #endif
00684
00685 bool tran{true}; // Should I transpose the Vandermonde matrix.
00686
00687 mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00688
00689 #if MTK_VERBOSE_LEVEL > 4
00690 std::cout << "aa_west_t =" << std::endl;
00691 std::cout << aa_west_t << std::endl;
00692 #endif
00693
00694 mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00695 (aa_west_t));
00696
00697 #if MTK_VERBOSE_LEVEL > 3
00698 std::cout << "qq_t = " << std::endl;
00699 std::cout << qq_t << std::endl;
00700 #endif
00701
00702 int kk_num_rows{num_bndy_coeffs_};
00703 int kk_num_cols{dim_null_};
00704
00705 mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00706
00707 // In the case of the gradient, even though we must solve for a null-space
00708 // of dimension 2, we must only extract ONE basis for the kernel.
00709 // We perform this extraction here:
00710
00711 int aux_{kk_num_rows - kk_num_cols};
00712 for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00713 aux_--;
00714 for (auto jj = 0; jj < kk_num_rows; jj++) {
00715 kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =
00716 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00717 }
00718 }
00719
00720 #if MTK_VERBOSE_LEVEL > 2
00721 std::cout << "kk =" << std::endl;
00722 std::cout << kk << std::endl;
00723 std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00724 std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00725 #endif
00726
00727 // Scale thus requesting that the last entries of the attained basis for the
00728 // null-space, adopt the pattern we require.
00729 // Essentially we will implement the following MATLAB pseudo-code:
00730 // scalers = kk(num_bndy_approx - (dim_null - 1):num_bndy_approx,:)\B
00731 // SK = kk*scalers
00732 // where SK is the scaled null-space.
00733
00734 // In this point, we almost have all the data we need correctly allocated
00735 // in memory. We will create the matrix iden_, and elements we wish to scale
00736 // in the kk array. Using the concept of the leading dimension, we could just
00737 // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00738 // GET how does it work. So I will just create a matrix with the content of
00739 // this array that we need, solve for the scalers and then scale the
00740 // whole kk:
00741
00742 // We will then create memory for that sub-matrix of kk (subk).
00743
00744 mtk::DenseMatrix subk(dim_null_, dim_null_);
00745
00746 auto zz = 0;
00747 for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00748 for (auto jj = 0; jj < dim_null_; jj++) {
00749 subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00750 }
00751 zz++;
00752 }
00753
00754 #if MTK_VERBOSE_LEVEL > 4

```

```

00763 std::cout << "subk =" << std::endl;
00764 std::cout << subk << std::endl;
00765 #endif
00766
00767 subk.Transpose();
00768
00769 #if MTK_VERBOSE_LEVEL > 4
00770 std::cout << "subk_t =" << std::endl;
00771 std::cout << subk << std::endl;
00772 #endif
00773
00774 bool padded{false};
00775 tran = false;
00776
00777 mtk::DenseMatrix iden(dim_null_, padded, tran);
00778
00779 #if MTK_VERBOSE_LEVEL > 4
00780 std::cout << "iden =" << std::endl;
00781 std::cout << iden << std::endl;
00782 #endif
00783
00784 // Solve the system to compute the scalars.
00785 // An example of the system to solve, for k = 8, is:
00786 //
00787 // subk*scalars = iden or
00788 //
00789 // | 0.386018 -0.0339244 -0.129478 | | 1 0 0 |
00790 // | -0.119774 0.0199423 0.0558632 |*scalars = | 0 1 0 |
00791 // | 0.0155708 -0.00349546 -0.00853182 | | 0 0 1 |
00792 //
00793 // Notice this is a nrhs = 3 system.
00794 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalars... they
00795 // will be stored in the created identity matrix.
00796 // Let us first transpose subk (because of LAPACK):
00797
00798 int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00799
00800 #ifdef MTK_PERFORM_PREVENTIONS
00801 if (!info) {
00802 std::cout << "System successfully solved!" <<
00803 std::endl;
00804 } else {
00805 std::cerr << "Something went wrong solving system! info = " << info <<
00806 std::endl;
00807 std::cerr << "Exiting..." << std::endl;
00808 return false;
00809 }
00810 std::cout << std::endl;
00811 #endif
00812
00813 #if MTK_VERBOSE_LEVEL > 4
00814 std::cout << "Computed scalars:" << std::endl;
00815 std::cout << iden << std::endl;
00816 #endif
00817
00818 // Multiply the two matrices to attain a scaled basis for null-space.
00819
00820 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00821
00822 #if MTK_VERBOSE_LEVEL > 4
00823 std::cout << "Rational basis for the null-space:" << std::endl;
00824 std::cout << rat_basis_null_space_ << std::endl;
00825 #endif
00826
00827 // At this point, we have a rational basis for the null-space, with the
00828 // pattern we need! :)
00829
00830 delete [] gg;
00831 gg = nullptr;
00832
00833 return true;
00834 }
00835
00836 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00837
00838
00839 mtk::Real *gg{}; // Generator vector for the first approximation.
00840
00841 try {
00842 gg = new mtk::Real[num_bndy_coeffs_];
00843 } catch (std::bad_alloc &memory_allocation_exception) {

```

```

00845 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00846 std::endl;
00847 std::cerr << memory_allocation_exception.what() << std::endl;
00848 }
00849 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00850
00851 #ifdef MTK_PRECISION_DOUBLE
00852 gg[1] = 1.0/2.0;
00853 #else
00854 gg[1] = 1.0f/2.0f;
00855 #endif
00856 for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00857 gg[ii] = gg[ii - 1] + mtk::kOne;
00858 }
00859
00860 #if MTK_VERBOSE_LEVEL > 3
00861 std::cout << "gg0 =" << std::endl;
00862 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00863 std::cout << std::setw(12) << gg[ii];
00864 }
00865 std::cout << std::endl << std::endl;
00866 #endif
00867
00868 // Allocate 2D array to store the collection of preliminary approximations.
00869 try {
00870 prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00871 } catch (std::bad_alloc &memory_allocation_exception) {
00872 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00873 std::endl;
00874 std::cerr << memory_allocation_exception.what() << std::endl;
00875 }
00876 memset(prem_apps_,
00877 mtk::kZero,
00878 sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00879
00880
00881 for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00882
00883 // Re-check new generator vector for every iteration except for the first.
00884 #if MTK_VERBOSE_LEVEL > 3
00885 if (ll > 0) {
00886 std::cout << "gg_" << ll << " =" << std::endl;
00887 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00888 std::cout << std::setw(12) << gg[ii];
00889 }
00890 std::cout << std::endl << std::endl;
00891 }
00892 #endif
00893
00894 bool transpose{false};
00895
00896 mtk::DenseMatrix aa(gg,
00897 num_bndy_coeffs_, order_accuracy_ + 1,
00898 transpose);
00899
00900 #if MTK_VERBOSE_LEVEL > 4
00901 std::cout << "aa_" << ll << " =" << std::endl;
00902 std::cout << aa << std::endl;
00903 #endif
00904
00905 mtk::Real *ob{};
00906
00907 auto ob_ld = num_bndy_coeffs_;
00908
00909 try {
00910 ob = new mtk::Real[ob_ld];
00911 } catch (std::bad_alloc &memory_allocation_exception) {
00912 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00913 std::endl;
00914 std::cerr << memory_allocation_exception.what() << std::endl;
00915 }
00916 memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00917
00918 ob[1] = mtk::kOne;
00919
00920 #if MTK_VERBOSE_LEVEL > 3
00921 std::cout << "ob =" << std::endl << std::endl;
00922 for (auto ii = 0; ii < ob_ld; ++ii) {
00923 std::cout << std::setw(12) << ob[ii] << std::endl;
00924 }
00925 #endif

```

```

00929 }
00930 std::cout << std::endl;
00931 #endif
00932
00933 // However, this is an under-determined system of equations. So we can not
00934 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00935 // our LAPACKAdapter class.
00936
00937 int info_{
00938 mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
00939 , ob_ld)};
00940
00941 #ifdef MTK_PERFORM_PREVENTIONS
00942 if (!info_) {
00943 std::cout << "System successfully solved!" << std::endl << std::endl;
00944 } else {
00945 std::cerr << "Error solving system! info = " << info_ << std::endl;
00946 return false;
00947 }
00948 #endif
00949
00950 #if MTK_VERBOSE_LEVEL > 3
00951 std::cout << "ob =" << std::endl;
00952 for (auto ii = 0; ii < ob_ld; ++ii) {
00953 std::cout << std::setw(12) << ob[ii] << std::endl;
00954 }
00955 std::cout << std::endl;
00956 #endif
00957
00958 // This implies a DAXPY operation. However, we must construct the arguments
00959 // for this operation.
00960
00961 // Save them into the ob_bottom array:
00962
00963 Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00964
00965 try {
00966 ob_bottom = new mtk::Real[dim_null_];
00967 } catch (std::bad_alloc &memory_allocation_exception) {
00968 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00969 std::endl;
00970 std::cerr << memory_allocation_exception.what() << std::endl;
00971 }
00972 memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00973
00974 for (auto ii = 0; ii < dim_null_; ++ii) {
00975 ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00976 }
00977
00978 #if MTK_VERBOSE_LEVEL > 3
00979 std::cout << "ob_bottom =" << std::endl;
00980 for (auto ii = 0; ii < dim_null_; ++ii) {
00981 std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00982 }
00983 std::cout << std::endl;
00984 #endif
00985
00986 // We must computed an scaled ob, sob, using the scaled null-space in
00987 // rat_basis_null_space_.
00988 // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00989 // or:
00990 // ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00991 // thus:
00992 // Y = a*A *x + b*Y (DAXPY).
00993
00994 #if MTK_VERBOSE_LEVEL > 4
00995 std::cout << "Rational basis for the null-space:" << std::endl;
00996 std::cout << rat_basis_null_space_ << std::endl;
00997 #endif
00998
00999 mtk::Real alpha{-mtk::kOne};
01000 mtk::Real beta{mtk::kOne};
01001
01002 mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
01003 ob_bottom, beta, ob);
01004
01005 #if MTK_VERBOSE_LEVEL > 3
01006 std::cout << "scaled ob:" << std::endl;
01007 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01008 std::cout << std::setw(12) << ob[ii] << std::endl;
01009 }
01010

```



```

01013 }
01014 std::cout << std::endl;
01015 #endif
01016
01017 // We save the recently scaled solution, into an array containing these.
01018 // We can NOT start building the pi matrix, simply because I want that part
01019 // to be separated since its construction depends on the algorithm we want
01020 // to implement.
01021
01022 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01023 prem_apps_[ii*num_bndy_approxs_ + 11] = ob[ii];
01024 }
01025
01026 // After the first iteration, simply shift the entries of the last
01027 // generator vector used:
01028 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01029 gg[ii]--;
01030 }
01031
01032 // Garbage collection for this loop:
01033 delete[] ob;
01034 ob = nullptr;
01035
01036 delete[] ob_bottom;
01037 ob_bottom = nullptr;
01038 } // End of: for (11 = 0; 11 < dim_null; 11++);
01039
01040 #if MTK_VERBOSE_LEVEL > 4
01041 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
01042 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01043 for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01044 std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
01045 }
01046 std::cout << std::endl;
01047 }
01048 std::cout << std::endl;
01049 #endif
01050
01051 delete[] gg;
01052 gg = nullptr;
01053
01054 return true;
01055 }
01056
01057 bool mtk::Grad1D::ComputeWeights() {
01058
01059 // Matrix to compute the weights as in the CRSA.
01060 mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
01061
01062 // Assemble the pi matrix using:
01063 // 1. The collection of scaled preliminary approximations.
01064 // 2. The collection of coefficients approximating at the interior.
01065 // 3. The scaled basis for the null-space.
01066
01067 // 1.1. Process array of scaled preliminary approximations.
01068
01069 // These are queued in scaled_solutions. Each one of these, will be a column
01070 // of the pi matrix:
01071 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01072 for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01073 pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01074 prem_apps_[ii*num_bndy_approxs_ + jj];
01075 }
01076 }
01077
01078 // 1.2. Add columns from known stencil approximating at the interior.
01079
01080 // However, these must be padded by zeros, according to their position in the
01081 // final pi matrix:
01082 auto mm = 1;
01083 for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
01084 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01085 auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
01086 (order_accuracy_/2 + 1)) + jj;
01087 pi.data()[de] = coeffs_interior_[ii];
01088 }
01089 ++mm;
01090 }
01091
01092 rat_basis_null_space_.OrderColMajor();
01093
01094

```

```

01095
01096 #if MTK_VERBOSE_LEVEL > 4
01097 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01098 std::cout << rat_basis_null_space_ << std::endl;
01099 #endif
01100
01101 // 1.3. Add final set of columns: rational basis for null-space.
01102
01103 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01104 jj < num_bndy_coeffs_ - 1; ++jj) {
01105 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01106 auto og =
01107 (jj - (dim_null_ + (order_accuracy_/2 + 1))) * num_bndy_coeffs_ + ii;
01108 auto de = ii * (2 * dim_null_ + (order_accuracy_/2 + 1)) + jj;
01109 pi.data()[de] = rat_basis_null_space_.data()[og];
01110 }
01111 }
01112
01113 #if MTK_VERBOSE_LEVEL > 4
01114 std::cout << "coeffs_interior_ =" << std::endl;
01115 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01116 std::cout << std::setw(12) << coeffs_interior_[ii];
01117 }
01118 std::cout << std::endl << std::endl;
01119 #endif
01120
01121 #if MTK_VERBOSE_LEVEL > 4
01122 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01123 std::cout << pi << std::endl;
01124 #endif
01125
01126 // This imposes the mimetic condition.
01127
01128 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01129
01130 try {
01131 hh = new mtk::Real[num_bndy_coeffs_];
01132 } catch (std::bad_alloc &memory_allocation_exception) {
01133 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01134 std::endl;
01135 std::cerr << memory_allocation_exception.what() << std::endl;
01136 }
01137 memset(hh, mtk::kZero, sizeof(hh[0]) * num_bndy_coeffs_);
01138
01139 hh[0] = -mtk::kOne;
01140 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01141 auto aux_xx = mtk::kZero;
01142 for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01143 aux_xx += coeffs_interior_[jj];
01144 }
01145 hh[ii] = -mtk::kOne * aux_xx;
01146 }
01147
01148 // That is, we construct a system, to solve for the weights.
01149
01150 // Once again we face the challenge of solving with LAPACK. However, for the
01151 // CRSA, this matrix PI is over-determined, since it has more rows than
01152 // unknowns. However, according to the theory, the solution to this system is
01153 // unique. We will use dgels_.
01154
01155 try {
01156 weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01157 } catch (std::bad_alloc &memory_allocation_exception) {
01158 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01159 std::endl;
01160 std::cerr << memory_allocation_exception.what() << std::endl;
01161 }
01162 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0]) * num_bndy_coeffs_);
01163
01164 int weights_ld{pi.num_cols() + 1};
01165
01166 // Preserve hh.
01167 std::copy(hh, hh + weights_ld, weights_cbs_);
01168
01169 pi.Transpose();
01170
01171 int info{
01172 mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01173 weights_cbs_, weights_ld)

```

```

01178 };
01179
01180 #ifdef MTK_PERFORM_PREVENTIONS
01181 if (!info) {
01182 std::cout << "System successfully solved!" << std::endl << std::endl;
01183 } else {
01184 std::cerr << "Error solving system! info = " << info << std::endl;
01185 return false;
01186 }
01187 #endif
01188
01189 #if MTK_VERBOSE_LEVEL > 3
01190 std::cout << "hh =" << std::endl;
01191 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01192 std::cout << std::setw(11) << hh[ii] << std::endl;
01193 }
01194 std::cout << std::endl;
01195 #endif
01196
01197 // Preserve the original weights for research.
01198
01199 try {
01200 weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01201 } catch (std::bad_alloc &memory_allocation_exception) {
01202 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01203 std::endl;
01204 std::cerr << memory_allocation_exception.what() << std::endl;
01205 }
01206 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01207
01208 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01209
01210 #if MTK_VERBOSE_LEVEL > 3
01211 std::cout << "weights_CRSA + lambda =" << std::endl;
01212 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01213 std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01214 }
01215 std::cout << std::endl;
01216 #endif
01217
01219
01220 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01221
01222 mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01223
01224 // 6.1. Insert preliminary approximations to first set of columns.
01225
01226 for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01227 for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01228 phi.data()[ii*(order_accuracy_) + jj] =
01229 prem_apps_[ii*num_bndy_approxs_ + jj];
01230 }
01231 }
01232
01233 // 6.2. Skip a column and negate preliminary approximations.
01234
01235 for (auto jj = 0; jj < order_accuracy_ + 1; jj++) {
01236 for (auto ii = 1; ii < num_bndy_approxs_; ii++) {
01237 auto de = (ii+order_accuracy_ - num_bndy_approxs_ + jj*order_accuracy_);
01238 auto og = (num_bndy_approxs_ - ii + (jj)*num_bndy_approxs_);
01239 phi.data()[de] = -pre_apps_[og];
01240 }
01241 }
01242
01243 // 6.3. Flip negative columns up-down.
01244
01245 for (auto ii = 0; ii < order_accuracy_/2; ii++) {
01246 for (auto jj = num_bndy_approxs_ + 1; jj < order_accuracy_; jj++) {
01247 auto aux = phi.data()[ii*order_accuracy_ + jj];
01248 phi.data()[ii*order_accuracy_ + jj] =
01249 phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01250 phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01251 }
01252 }
01253
01254 // 6.4. Insert stencil.
01255
01256 auto mm = 0;
01257 for (auto jj = num_bndy_approxs_; jj < num_bndy_approxs_ + 1; jj++) {
01258 for (auto ii = 0; ii < order_accuracy_ + 1; ii++) {

```

```

01261 if (ii == 0) {
01262 phi.data()[jj] = 0.0;
01263 } else {
01264 phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior_[ii - 1];
01265 }
01266 }
01267 mm++;
01268 }
01269
01270 #if MTK_VERBOSE_LEVEL > 4
01271 std::cout << "phi =" << std::endl;
01272 std::cout << phi << std::endl;
01273 #endif
01274
01275
01276
01277 mtk::Real *lamed{}; // Used to build big lambda.
01278
01279 try {
01280 lamed = new mtk::Real[num_bndy_approxs_ - 1];
01281 } catch (std::bad_alloc &memory_allocation_exception) {
01282 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01283 std::endl;
01284 std::cerr << memory_allocation_exception.what() << std::endl;
01285 }
01286 memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approxs_ - 1));
01287
01288 for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01289 lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01290 }
01291
01292 #if MTK_VERBOSE_LEVEL > 3
01293 std::cout << "lamed =" << std::endl;
01294 for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01295 std::cout << std::setw(12) << lamed[ii] << std::endl;
01296 }
01297 std::cout << std::endl;
01298 #endif
01299
01300 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01301 mtk::Real temp = mtk::kZero;
01302 for (auto jj = 0; jj < num_bndy_approxs_ - 1; ++jj) {
01303 temp = temp +
01304 lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01305 }
01306 hh[ii] = hh[ii] - temp;
01307 }
01308
01309 #if MTK_VERBOSE_LEVEL > 3
01310 std::cout << "big_lambda =" << std::endl;
01311 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01312 std::cout << std::setw(12) << hh[ii] << std::endl;
01313 }
01314 std::cout << std::endl;
01315 #endif
01316
01317 #ifdef MTK_VERBOSE_WEIGHTS
01318 int copy_result{1};
01319 #else
01320 int copy_result{};
01321 #endif
01322
01323 int minrow{std::numeric_limits<int>::infinity()};
01324
01325 mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01326 order_accuracy_)};
01327 mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01328
01329 mtk::Real normerr_; // Norm of the error for the solution on each row.
01330
01331 #ifdef MTK_VERBOSE_WEIGHTS
01332 std::ofstream table("grad_ld_" + std::to_string(order_accuracy_) +
01333 "_weights.tex");
01334
01335 table << "\\begin{tabular}[c]{c}";
01336 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01337 table << 'c';
01338 }
01339 table << "c}\\toprule\\nRow & ";
01340 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01341 table << "$ q_{ " + std::to_string(ii) + " }$ & ";

```

```

01343 }
01344 table << " Relative error \\\n\\midrule\n";
01345 #endif
01346
01347 for(auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01348 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01349 order_accuracy_ + 1,
01350 order_accuracy_,
01351 order_accuracy_,
01352 hh,
01353 weights_cbs_,
01354 row_,
01355 mimetic_threshold_,
01356 copy_result);
01357 mtk::Real aux{normerr_/norm};
01358
01359 #if MTK_VERBOSE_LEVEL > 2
01360 std::cout << "Relative norm: " << aux << " " << std::endl;
01361 std::cout << std::endl;
01362 #endif
01363
01364 if (aux < minnorm) {
01365 minnorm = aux;
01366 minrow_ = row_;
01367 }
01368
01369 #ifdef MTK_VERBOSE_WEIGHTS
01370 table << std::to_string(row_ + 1) << " & ";
01371 if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01372 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01373 table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01374 }
01375 table << std::to_string(aux) << " \\\n" << std::endl;
01376 } else {
01377 table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
" }{c}{\\emptyset} & ";
01378 table << " - \\\n" << std::endl;
01379 }
01380 }
01381 #endif
01382 }
01383
01384 #ifdef MTK_VERBOSE_WEIGHTS
01385 table << "\\midrule" << std::endl;
01386 table << "CRS weights:";
01387 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01388 table << " & " << std::to_string(weights_crs_[ii - 1]);
01389 }
01390 table << " & - \\\n\\bottomrule\n\\end{tabular}" << std::endl;
01391 table.close();
01392 #endif
01393
01394 #if MTK_VERBOSE_LEVEL > 3
01395 std::cout << "weights_CBSA + lambda (after brute force search):" <<
std::endl;
01396 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01397 std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01398 }
01399 std::cout << std::endl;
01400 #endif
01401
01402
01403 // After we know which row yields the smallest relative norm that row is
01404 // chosen to be the objective function and the result of the optimizer is
01405 // chosen to be the new weights_.
01406
01407
01408 #if MTK_VERBOSE_LEVEL > 2
01409 std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
minrow_ + 1 << std::endl;
01410 std::cout << std::endl;
01411 #endif
01412
01413 copy_result = 1;
01414 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01417 order_accuracy_ + 1,
01418 order_accuracy_,
01419 order_accuracy_,
01420 hh,
01421 weights_cbs_,
01422 minrow_,

```

```

01423 mimetic_threshold_,
01424 copy_result);
01425 mtk::Real aux_{normerr_/norm};
01426 #if MTK_VERBOSE_LEVEL > 2
01427 std::cout << "Relative norm: " << aux_ << std::endl;
01428 std::cout << std::endl;
01429 #endif
01430
01431 delete [] lamed;
01432 lamed = nullptr;
01433 }
01434
01435 delete [] hh;
01436 hh = nullptr;
01437
01438 return true;
01439 }
01440
01441 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01442
01443 #if MTK_VERBOSE_LEVEL > 3
01444 std::cout << "weights_* + lambda =" << std::endl;
01445 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01446 std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01447 }
01448 std::cout << std::endl;
01449 #endif
01450
01451 mtk::Real *lambda{}; // Collection of bottom values from weights_.
01452
01453 try {
01454 lambda = new mtk::Real[dim_null_];
01455 } catch (std::bad_alloc &memory_allocation_exception) {
01456 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01457 std::endl;
01458 std::cerr << memory_allocation_exception.what() << std::endl;
01459 }
01460 memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01461
01462 for (auto ii = 0; ii < dim_null_; ++ii) {
01463 lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01464 }
01465
01466 #if MTK_VERBOSE_LEVEL > 3
01467 std::cout << "lambda =" << std::endl;
01468 for (auto ii = 0; ii < dim_null_; ++ii) {
01469 std::cout << std::setw(12) << lambda[ii] << std::endl;
01470 }
01471 std::cout << std::endl;
01472 #endif
01473
01474 mtk::Real *alpha{}; // Collection of alpha values.
01475
01476 try {
01477 alpha = new mtk::Real[dim_null_];
01478 } catch (std::bad_alloc &memory_allocation_exception) {
01479 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01480 std::endl;
01481 std::cerr << memory_allocation_exception.what() << std::endl;
01482 }
01483 memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01484
01485 for (auto ii = 0; ii < dim_null_; ++ii) {
01486 alpha[ii] = lambda[ii]/weights_cbs_[ii];
01487 }
01488
01489 #if MTK_VERBOSE_LEVEL > 3
01490 std::cout << "alpha =" << std::endl;
01491 for (auto ii = 0; ii < dim_null_; ++ii) {
01492 std::cout << std::setw(12) << alpha[ii] << std::endl;
01493 }
01494 std::cout << std::endl;
01495 #endif
01496
01497 try {
01498 mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01499 } catch (std::bad_alloc &memory_allocation_exception) {
01500 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```

```

01507 std::endl;
01508 std::cerr << memory_allocation_exception.what() << std::endl;
01509 }
01510 memset(mim_bndy_,
01511 mtk::kZero,
01512 sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01513
01514 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01515 for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {
01516 mim_bndy_[ii*num_bndy_approxs_ + jj] =
01517 prem_apps_[ii*num_bndy_approxs_ + jj] +
01518 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01519 }
01520 }
01521
01522 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01523 mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01524 prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01525 }
01526
01527 #if MTK_VERBOSE_LEVEL > 4
01528 std::cout << "Collection of mimetic approximations:" << std::endl;
01529 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01530 for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01531 std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01532 }
01533 std::cout << std::endl;
01534 }
01535 std::cout << std::endl;
01536 #endif
01537
01538 delete[] lambda;
01539 lambda = nullptr;
01540
01541 delete[] alpha;
01542 alpha = nullptr;
01543
01544 return true;
01545 }
01546
01547 bool mtk::Grad1D::AssembleOperator(void) {
01548
01549 // The output array will have this form:
01550 // 1. The first entry of the array will contain the used order kk.
01551 // 2. The second entry of the array will contain the collection of
01552 // approximating coefficients for the interior of the grid.
01553 // 3. The third entry will contain a collection of weights.
01554 // 4. The next dim_null - 1 entries will contain the collections of
01555 // approximating coefficients for the west boundary of the grid.
01556
01557 gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01558 num_bndy_approxs_*num_bndy_coeffs_;
01559
01560 #if MTK_VERBOSE_LEVEL > 2
01561 std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01562 #endif
01563
01564 try {
01565 gradient_ = new mtk::Real[gradient_length_];
01566 } catch (std::bad_alloc &memory_allocation_exception) {
01567 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01568 std::endl;
01569 std::cerr << memory_allocation_exception.what() << std::endl;
01570 }
01571 memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01572
01573
01574
01575 gradient_[0] = order_accuracy_;
01576
01577
01578 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01579 gradient_[ii + 1] = coeffs_interior_[ii];
01580 }
01581
01582
01583
01584
01585 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01586 gradient_[(order_accuracy_ + 1) + ii] = weights_cbs_[ii];
01587 }
01588
01589
01590
01591 int offset{2*order_accuracy_ + 1};
01592
01593

```

```

01594
01595 int aux {}; // Auxiliary variable.
01596
01597 if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01598 for (auto ii = 0; ii < num_bndy_approxs_ ; ii++) {
01599 for (auto jj = 0; jj < num_bndy_coeffs_ ; jj++) {
01600 gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01601 aux++;
01602 }
01603 }
01604 } else {
01605 gradient_[offset + 0] = prem_apps_[0];
01606 gradient_[offset + 1] = prem_apps_[1];
01607 gradient_[offset + 2] = prem_apps_[2];
01608 }
01609
01610 #if MTK_VERBOSE_LEVEL > 1
01611 std::cout << "1D " << order_accuracy_ << "-order grad built!" << std::endl;
01612 std::cout << std::endl;
01613 #endif
01614
01615 return true;
01616 }

```

## 18.93 src/mtk\_grad\_2d.cc File Reference

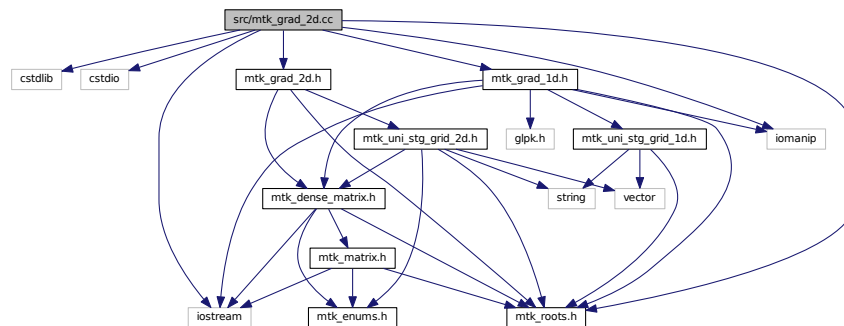
Implements the class Grad2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_2d.h"

```

Include dependency graph for mtk\_grad\_2d.cc:



### 18.93.1 Detailed Description

This class implements a 2D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).



## Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_2d.cc](#).

## 18.94 mtk\_grad\_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_2d.h"
00066
00067 mtk::Grad2D::Grad2D():
00068 order_accuracy_(),
00069 mimetic_threshold_() {}
00070
00071 mtk::Grad2D::Grad2D(const Grad2D &grad):
00072 order_accuracy_(grad.order_accuracy_),
00073 mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad2D::~~Grad2D() {}
00076
00077 bool mtk::Grad2D::ConstructGrad2D(const

```

```

 mtk::UniStgGrid2D &grid,
00078 int order_accuracy,
00079 mtk::Real mimetic_threshold) {
00080
00081 int num_cells_x = grid.num_cells_x();
00082 int num_cells_y = grid.num_cells_y();
00083
00084 int mx = num_cells_x + 1; // Gx vertical dimension
00085 int nx = num_cells_x + 2; // Gx horizontal dimension
00086 int my = num_cells_y + 1; // Gy vertical dimension
00087 int ny = num_cells_y + 2; // Gy horizontal dimension
00088
00089 mtk::Grad1D grad;
00090
00091 bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00092
00093 #ifdef MTK_PERFORM_PREVENTIONS
00094 if (!info) {
00095 std::cerr << "Mimetic grad could not be built." << std::endl;
00096 return info;
00097 }
00098 #endif
00099
00100 auto west = grid.west_bndy();
00101 auto east = grid.east_bndy();
00102 auto south = grid.south_bndy();
00103 auto north = grid.east_bndy();
00104
00105 mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00106 mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00107
00108 mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00109 mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00110
00111 bool padded{true};
00112 bool transpose{true};
00113
00114 mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00115 mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00116
00117 mtk::DenseMatrix gxy(mtk::DenseMatrix::Kron(tiy, Gx));
00118 mtk::DenseMatrix gyx(mtk::DenseMatrix::Kron(Gy, tix));
00119
00120 #if MTK_VERBOSE_LEVEL > 2
00121 std::cout << "Gx: " << mx << " by " << nx << std::endl;
00122 std::cout << "Transpose Iy: " << num_cells_y << " by " << ny << std::endl;
00123 std::cout << "Gy: " << my << " by " << ny << std::endl;
00124 std::cout << "Transpose Ix: " << num_cells_x << " by " << nx << std::endl;
00125 std::cout << "Grad 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00126 nx*ny <<std::endl;
00127 #endif
00128
00129 mtk::DenseMatrix g2d(mx*num_cells_y + my*num_cells_x, nx*ny);
00130
00131 for(auto ii = 0; ii < nx*ny; ii++) {
00132 for(auto jj = 0; jj < mx*num_cells_y; jj++) {
00133 g2d.SetValue(jj,ii, gxy.GetValue(jj,ii));
00134 }
00135 for(auto kk = 0; kk < my*num_cells_x; kk++) {
00136 g2d.SetValue(kk + mx*num_cells_y, ii, gyx.GetValue(kk,ii));
00137 }
00138 }
00139
00140 gradient_ = g2d;
00141
00142 return info;
00143 }
00144
00145 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix() const {
00146
00147 return gradient_;
00148 }

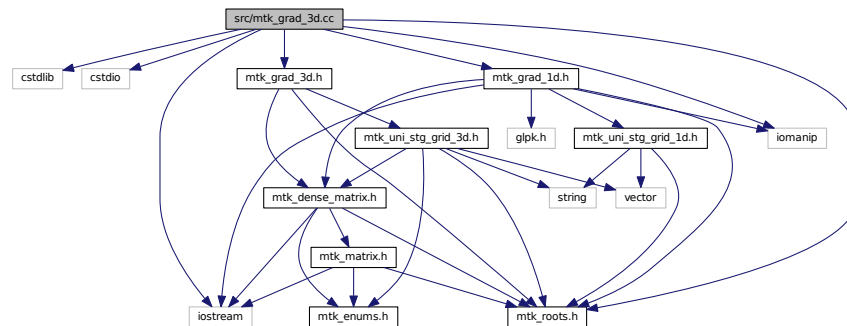
```

## 18.95 src/mtk\_grad\_3d.cc File Reference

Implements the class Grad3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_3d.h"
```

Include dependency graph for mtk\_grad\_3d.cc:



### 18.95.1 Detailed Description

This class implements a 3D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_3d.cc](#).

## 18.96 mtk\_grad\_3d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
```

```

00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_3d.h"
00066
00067 mtk::Grad3D::Grad3D():
00068 order_accuracy_(),
00069 mimetic_threshold_() {}
00070
00071 mtk::Grad3D::Grad3D(const Grad3D &grad):
00072 order_accuracy_(grad.order_accuracy_),
00073 mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad3D::~Grad3D() {}
00076
00077 bool mtk::Grad3D::ConstructGrad3D(const
 mtk::UniStgGrid3D &grid,
 int order_accuracy,
 mtk::Real mimetic_threshold) {
00078
00079
00080
00081 int num_cells_x = grid.num_cells_x();
00082 int num_cells_y = grid.num_cells_y();
00083 int num_cells_z = grid.num_cells_z();
00084
00085 int mx = num_cells_x + 1; // Gx vertical dimension.
00086 int nx = num_cells_x + 2; // Gx horizontal dimension.
00087 int my = num_cells_y + 1; // Gy vertical dimension.
00088 int ny = num_cells_y + 2; // Gy horizontal dimension.
00089 int mz = num_cells_z + 1; // Gz vertical dimension.
00090 int nz = num_cells_z + 2; // Gz horizontal dimension.
00091
00092 mtk::Grad1D grad;
00093
00094 bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00095
00096 #ifdef MTK_PERFORM_PREVENTIONS
00097 if (!info) {
00098 std::cerr << "Mimetic grad could not be built." << std::endl;
00099 return info;
00100 }
00101 #endif
00102
00103 auto west = grid.west_bndy();
00104 auto east = grid.east_bndy();
00105 auto south = grid.south_bndy();
00106 auto north = grid.north_bndy();
00107 auto bottom = grid.bottom_bndy();
00108 auto top = grid.top_bndy();
00109
00110 mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00111 mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00112 mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z);

```

```

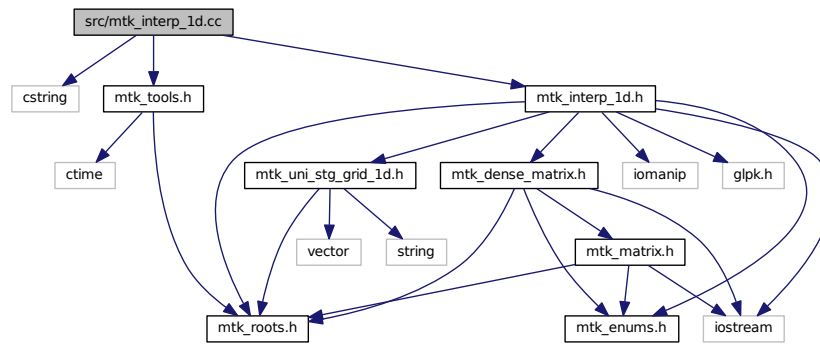
00113
00114 mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00115 mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00116 mtk::DenseMatrix Gz(grad.ReturnAsDenseMatrix(grid_z));
00117
00118 bool padded{true};
00119 bool transpose{true};
00120
00121 mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00122 mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00123 mtk::DenseMatrix tiz(num_cells_z, padded, transpose);
00124
00126
00127 mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(tiz, tiy));
00128 mtk::DenseMatrix gx(mtk::DenseMatrix::Kron(aux1, Gx));
00129
00131
00132 mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(tiz, Gy));
00133 mtk::DenseMatrix gy(mtk::DenseMatrix::Kron(aux2, tix));
00134
00136
00137 mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Gz, tiy));
00138 mtk::DenseMatrix gz(mtk::DenseMatrix::Kron(aux3, tix));
00139
00140 #if MTK_VERBOSE_LEVEL > 2
00141 std::cout << "Gx: " << mx << " by " << nx << std::endl;
00142 std::cout << "Transpose Ix: " << num_cells_x << " by " << nx << std::endl;
00143 std::cout << "Gy: " << my << " by " << ny << std::endl;
00144 std::cout << "Transpose Iy: " << num_cells_y << " by " << ny << std::endl;
00145 std::cout << "Gz: " << mz << " by " << nz << std::endl;
00146 std::cout << "Transpose Iz: " << num_cells_z << " by " << nz << std::endl;
00147 #endif
00148
00150
00151 int total_rows{mx*num_cells_y*num_cells_z +
00152 num_cells_x*my*num_cells_z +
00153 num_cells_x*num_cells_y*mz};
00154 int total_cols{nx*ny*nz};
00155
00156 #if MTK_VERBOSE_LEVEL > 2
00157 std::cout << "Grad 3D: " << total_rows << " by " << total_cols << std::endl;
00158 #endif
00159
00160 mtk::DenseMatrix g3d(total_rows, total_cols);
00161
00162 for(auto ii = 0; ii < total_cols; ii++) {
00163 for(auto jj = 0; jj < mx*num_cells_y*num_cells_z; jj++) {
00164 g3d.SetValue(jj,ii, gx.GetValue(jj,ii));
00165 }
00166
00167 int offset = mx*num_cells_y*num_cells_z;
00168
00169 for(auto kk = 0; kk < num_cells_x*my*num_cells_z; kk++) {
00170 g3d.SetValue(kk + offset, ii, gy.GetValue(kk,ii));
00171 }
00172
00173 offset += num_cells_x*my*num_cells_z;
00174
00175 for(auto ll = 0; ll < num_cells_x*num_cells_y*mz; ll++) {
00176 g3d.SetValue(ll + offset, ii, gz.GetValue(ll,ii));
00177 }
00178 }
00179
00180 gradient_ = g3d;
00181
00182 return info;
00183 }
00184
00185 mtk::DenseMatrix mtk::Grad3D::ReturnAsDenseMatrix() const {
00186
00187 return gradient_;
00188 }

```

## 18.97 src/mtk\_interp\_1d.cc File Reference

Includes the implementation of the class Interp1D.

```
#include <cstring>
#include "mtk_tools.h"
#include "mtk_interp_1d.h"
Include dependency graph for mtk_interp_1d.cc:
```



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::Interp1D &in)`

### 18.97.1 Detailed Description

This class implements a 1D interpolation operator.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu  
 : Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file [mtk\\_interp\\_1d.cc](#).

## 18.98 mtk\_interp\_1d.cc

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
```

```

00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.cs.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.cs.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #include <cstring>
00059
00060 #include "mtk_tools.h"
00061
00062 #include "mtk_interp_1d.h"
00063
00064 namespace mtk {
00065
00066 std::ostream& operator <<(std::ostream &stream, mtk::Interp1D &in) {
00067
00068 stream << "coeffs_interior_[1:" << in.order_accuracy_ << "] = ";
00069 for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00070 stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00071 }
00072 stream << std::endl;
00073
00074 return stream;
00075 }
00076
00077 mtk::Interp1D::Interp1D():
00078 dir_interp_(mtk::SCALAR_TO_VECTOR),
00079 order_accuracy_(mtk::kDefaultOrderAccuracy),
00080 coeffs_interior_(nullptr) {}
00081
00082 mtk::Interp1D::Interp1D(const Interp1D &interp):
00083 dir_interp_(interp.dir_interp_),
00084 order_accuracy_(interp.order_accuracy_),
00085 coeffs_interior_(interp.coeffs_interior_) {}
00086
00087 mtk::Interp1D::~Interp1D() {
00088 delete[] coeffs_interior_;
00089 coeffs_interior_ = nullptr;
00090 }
00091
00092 bool mtk::Interp1D::ConstructInterp1D(int order_accuracy,
00093 mtk::DirInterp dir) {
00094
00095 #if MTK_PERFORM_PREVENTIONS
00096 mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00097 mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00098 mtk::Tools::Prevent(dir < mtk::SCALAR_TO_VECTOR &&

```

```

00102 dir > mtk::VECTOR_TO_SCALAR,
00103 __FILE__, __LINE__, __func__);
00104 #endif
00105
00106 #if MTK_VERBOSE_LEVEL > 2
00107 std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00108 #endif
00109
00110 order_accuracy_ = order_accuracy;
00111
00112 try {
00113 coeffs_interior_ = new mtk::Real[order_accuracy_];
00114 } catch (std::bad_alloc &memory_allocation_exception) {
00115 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00116 std::endl;
00117 std::cerr << memory_allocation_exception.what() << std::endl;
00118 }
00119 memset(coeffs_interior_,
00120 mtk::kZero,
00121 sizeof(coeffs_interior_[0])*order_accuracy_);
00122
00123 for (int ii = 0; ii < order_accuracy_; ++ii) {
00124 coeffs_interior_[ii] = mtk::kOne;
00125 }
00126 return true;
00127 }
00128
00129 mtk::Real *mtk::Interp1D::coeffs_interior() const {
00130 return coeffs_interior_;
00131 }
00132
00133 mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix(
00134 const UniStgGrid1D &grid) const {
00135 int nn(grid.num_cells_x()); // Number of cells on the grid.
00136
00137 #if MTK_PERFORM_PREVENTIONS
00138 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00139 #endif
00140
00141 int gg_num_rows{}; // Number of rows.
00142 int gg_num_cols{}; // Number of columns.
00143
00144 if (dir_interp_ == mtk::SCALAR_TO_VECTOR) {
00145 gg_num_rows = nn + 1;
00146 gg_num_cols = nn + 2;
00147 } else {
00148 gg_num_rows = nn + 2;
00149 gg_num_cols = nn + 1;
00150 }
00151
00152 // Output matrix featuring sizes for gradient operators.
00153 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00154
00155 out.SetValue(0, 0, mtk::kOne);
00156
00157 for (auto ii = 1; ii < gg_num_rows - 1; ++ii) {
00158 for (auto jj = ii; jj < order_accuracy_ + ii; ++jj) {
00159 out.SetValue(ii, jj, mtk::kOne/order_accuracy_);
00160 }
00161 }
00162
00163 out.SetValue(gg_num_rows - 1, gg_num_cols - 1, mtk::kOne);
00164
00165 return out;
00166 }

```

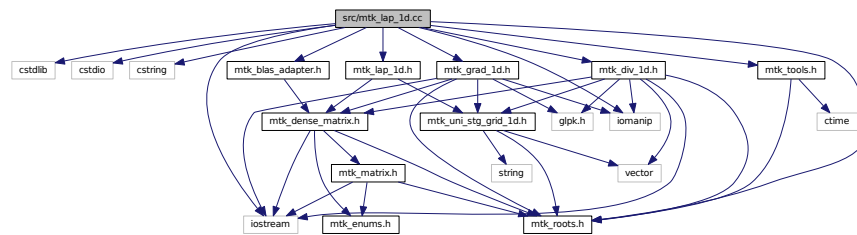


## 18.99 src/mtk\_lap\_1d.cc File Reference

Includes the implementation of the class Lap1D.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
```

Include dependency graph for mtk\_lap\_1d.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::Lap1D &in)`

### 18.99.1 Detailed Description

This class implements a 1D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_1d.cc](#).

## 18.100 mtk\_lap\_1d.cc

00001

```

00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059 #include <cstring>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_tools.h"
00066 #include "mtk_blas_adapter.h"
00067 #include "mtk_grad_ld.h"
00068 #include "mtk_div_ld.h"
00069 #include "mtk_lap_ld.h"
00070
00071 namespace mtk {
00072
00073 std::ostream& operator <<(std::ostream &stream, mtk::Lap1D &in) {
00074
00075 stream << "laplacian_[0] = " << in.laplacian_[0] << std::endl << std::endl;
00076
00077 stream << "laplacian_[1:] = " << 2*in.order_accuracy_ - 1 << " = " <<
00078 std::endl << std::endl;
00079 for (auto ii = 1; ii <= (2*in.order_accuracy_ - 1); ++ii) {
00080 stream << std::setw(13) << in.laplacian_[ii] << " ";
00081 }
00082 stream << std::endl << std::endl;
00083
00084 auto offset = 1 + (2*in.order_accuracy_ - 1);
00085
00086 stream << "laplacian_[" << offset << ":" << offset +
00087 (in.order_accuracy_ - 1)*(2*in.order_accuracy_ - 1) << "] = " <<
00088 std::endl << std::endl;

```

```

00095
00096 for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00097 for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00098 stream << std::setw(13) <<
00099 in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj];
00100 }
00101 stream << std::endl;
00102 }
00103
00104 return stream;
00105 }
00106 }
00107
00108 mtk::Lap1D::Lap1D():
00109 order_accuracy_(mtk::kDefaultOrderAccuracy),
00110 laplacian_length_(),
00111 delta_(mtk::kZero),
00112 mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00113
00114 mtk::Lap1D::~Lap1D() {
00115
00116 delete [] laplacian_;
00117 laplacian_ = nullptr;
00118 }
00119
00120 int mtk::Lap1D::order_accuracy() const {
00121
00122 return order_accuracy_;
00123 }
00124
00125 mtk::Real mtk::Lap1D::mimetic_threshold() const {
00126
00127 return mimetic_threshold_;
00128 }
00129
00130 mtk::Real mtk::Lap1D::delta() const {
00131
00132 return delta_;
00133 }
00134
00135 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00136 mtk::Real mimetic_threshold) {
00137
00138 #ifdef MTK_PERFORM_PREVENTIONS
00139 mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00140 mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00141 mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00142 __FILE__, __LINE__, __func__);
00143
00144 if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00145 std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00146 }
00147
00148 std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00149 std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00150 #endif
00151
00152 order_accuracy_ = order_accuracy;
00153 mimetic_threshold_ = mimetic_threshold;
00154
00155 mtk::Grad1D grad; // Mimetic gradient.
00156
00157 bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00158
00159 #ifdef MTK_PERFORM_PREVENTIONS
00160 if (!info) {
00161 std::cerr << "Mimetic grad could not be built." << std::endl;
00162 return false;
00163 }
00164 #endif
00165
00166 mtk::Div1D div; // Mimetic divergence.
00167
00168 info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00169
00170 #ifdef MTK_PERFORM_PREVENTIONS
00171 if (!info) {
00172 std::cerr << "Mimetic div could not be built." << std::endl;
00173 return false;
00174 }
00175

```

```

00178 #endif
00179
00181
00182 // Since these are mimetic operator, we must multiply the matrices arising
00183 // from both the divergence and the Laplacian, in order to get the
00184 // approximating coefficients for the Laplacian operator.
00185
00186 // However, we must choose a grid that implied a step size of 1, so to get
00187 // the approximating coefficients, without being affected from the
00188 // normalization with respect to the grid (dimensionless).
00189
00190 // Also, the grid must be of the minimum size to support the requested order
00191 // of accuracy. We must please the divergence for this!
00192
00193 mtk::UniStgGrid1D aux(mtk::kZero,
00194 (mtk::Real) 3*order_accuracy_ - 1,
00195 3*order_accuracy_ - 1);
00196
00197 #if MTK_VERBOSE_LEVEL > 2
00198 std::cout << "aux =" << std::endl;
00199 std::cout << aux << std::endl;
00200 std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00201 std::cout << std::endl;
00202 #endif
00203
00204 mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00205
00206 #if MTK_VERBOSE_LEVEL > 4
00207 std::cout << "grad_m =" << std::endl;
00208 std::cout << grad_m << std::endl;
00209 #endif
00210
00211 mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux));
00212
00213 #if MTK_VERBOSE_LEVEL > 4
00214 std::cout << "div_m =" << std::endl;
00215 std::cout << div_m << std::endl;
00216 #endif
00217
00221
00222 mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00223
00224 lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00225
00226 #if MTK_VERBOSE_LEVEL > 4
00227 std::cout << "lap =" << std::endl;
00228 std::cout << lap << std::endl;
00229 #endif
00230
00232
00233
00234
00235 // The output array will have this form:
00236 // 1. The first entry of the array will contain the used order kk.
00237 // 2. The second entry of the array will contain the collection of
00238 // approximating coefficients for the interior of the grid.
00239 // 3. The next entries will contain the collections of approximating
00240 // coefficients for the west boundary of the grid.
00241
00242 laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00243 (order_accuracy_ - 1)*(2*order_accuracy_);
00244
00245 #if MTK_VERBOSE_LEVEL > 2
00246 std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00247 std::cout << std::endl;
00248 #endif
00249
00250 try {
00251 laplacian_ = new mtk::Real[laplacian_length_];
00252 } catch (std::bad_alloc &memory_allocation_exception) {
00253 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00254 std::endl;
00255 std::cerr << memory_allocation_exception.what() << std::endl;
00256 }
00257 memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00258
00260
00261 laplacian_[0] = order_accuracy_;
00262
00265
00266 for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00267 laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);

```

```

00268 }
00269
00271
00272 auto offset = 1 + (2*order_accuracy_ - 1);
00273
00274 for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00275 for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00276 laplacian_[offset + ii*(2*order_accuracy_) + jj] =
00277 lap.GetValue(1 + ii, jj);
00278 }
00279 }
00280
00281 delta_ = mtk::kZero;
00282
00283 return true;
00284 }
00285
00286 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(
00287 const UniStgGrid1D &grid) const {
00288
00289 int nn{grid.num_cells_x()}; // Number of cells on the grid.
00290
00291 #ifdef MTK_PERFORM_PREVENTIONS
00292 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00293 mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00294 #endif
00295
00296 mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00297
00298 delta_ = grid.delta_x();
00299
00300 mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
00301 dx^2.
00302
00303
00304 auto offset = (1 + 2*order_accuracy_ - 1);
00305
00306 for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00307 for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00308 lap.SetValue(1 + ii,
00309 jj,
00310 idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00311 }
00312 }
00313
00315
00316 offset = 1 + (order_accuracy_ - 1);
00317
00318 int kk{1};
00319 for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00320 int mm{1};
00321 for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00322 lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00323 mm = mm + 1;
00324 }
00325 kk = kk + 1;
00326 }
00327
00329
00330 offset = (1 + 2*order_accuracy_ - 1);
00331
00332 auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00333
00334 auto ll = 1;
00335 auto rr = 1;
00336 for (auto ii = nn; ii > aux - 1; --ii) {
00337 auto cc = 0;
00338 for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00339 lap.SetValue(ii, jj, lap.GetValue(rr, cc));
00340 ++ll;
00341 ++cc;
00342 }
00343 rr++;
00344 }
00345
00352
00353 return lap;
00354 }
00355
00356 const mtk::Real* mtk::Lap1D::data(const UniStgGrid1D &grid) const {
00357

```

```

00358 mtk::DenseMatrix tmp;
00359
00360 tmp = ReturnAsDenseMatrix(grid);
00361
00362 return tmp.data();
00363 }

```

## 18.101 src/mtk\_lap\_2d.cc File Reference

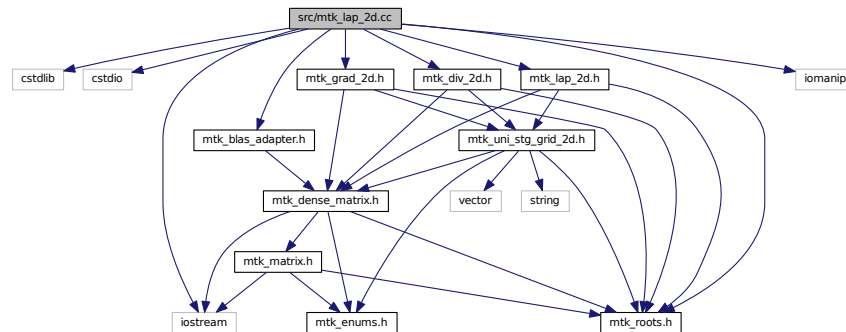
Includes the implementation of the class Lap2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_lap_2d.h"

```

Include dependency graph for mtk\_lap\_2d.cc:



### 18.101.1 Detailed Description

This class implements a 2D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_2d.cc](#).

## 18.102 mtk\_lap\_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.

```

```

00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_2d.h"
00066 #include "mtk_div_2d.h"
00067 #include "mtk_lap_2d.h"
00068
00069 mtk::Lap2D::Lap2D(): order_accuracy_(), mimetic_threshold_() {}
00070
00071 mtk::Lap2D::Lap2D(const Lap2D &lap):
00072 order_accuracy_(lap.order_accuracy_),
00073 mimetic_threshold_(lap.mimetic_threshold_) {}
00074
00075 mtk::Lap2D::~~Lap2D() {}
00076
00077 bool mtk::Lap2D::ConstructLap2D(const
 mtk::UniStgGrid2D &grid,
00078 int order_accuracy,
00079 mtk::Real mimetic_threshold) {
00080
00081 mtk::Grad2D gg;
00082 mtk::Div2D dd;
00083
00084 bool info{gg.ConstructGrad2D(grid, order_accuracy, mimetic_threshold)};
00085
00086 #ifdef MTK_PERFORM_PREVENTIONS
00087 if (!info) {
00088 std::cerr << "Mimetic lap could not be built." << std::endl;
00089 return info;
00090 }
00091 #endif
00092
00093 info = dd.ConstructDiv2D(grid, order_accuracy, mimetic_threshold);

```

```

00094
00095 #ifdef MTK_PERFORM_PREVENTIONS
00096 if (!info) {
00097 std::cerr << "Mimetic div could not be built." << std::endl;
00098 return info;
00099 }
00100 #endif
00101
00102 mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00103 mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00104
00105 laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
00106
00107 return info;
00108 }
00109
00110 mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix() const {
00111
00112 return laplacian_;
00113 }
00114
00115 mtk::Real *mtk::Lap2D::data() const {
00116
00117 return laplacian_.data();
00118 }

```

## 18.103 src/mtk\_lap\_3d.cc File Reference

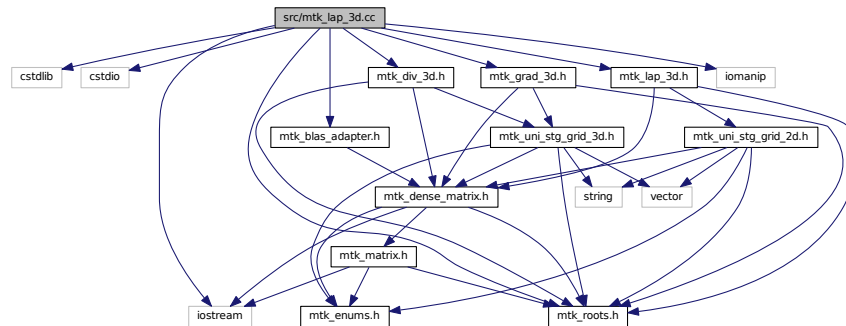
Includes the implementation of the class Lap3D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"

```

Include dependency graph for mtk\_lap\_3d.cc:



### 18.103.1 Detailed Description

This class implements a 3D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).



## Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_3d.cc](#).

## 18.104 mtk\_lap\_3d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_3d.h"
00066 #include "mtk_div_3d.h"
00067 #include "mtk_lap_3d.h"
00068
00069 mtk::UniStgGrid3D mtk::Lap3D::operator*(const
 mtk::UniStgGrid3D &grid) const {
00070
00071 mtk::UniStgGrid3D out;
00072
00073 return out;
00074 }
00075
00076 mtk::Lap3D::Lap3D(): order_accuracy_(), mimetic_threshold_() {}

```

```

00077
00078 mtk::Lap3D::Lap3D(const Lap3D &lap):
00079 order_accuracy_(lap.order_accuracy_),
00080 mimetic_threshold_(lap.mimetic_threshold_) {}
00081
00082 mtk::Lap3D::~~Lap3D() {}
00083
00084 bool mtk::Lap3D::ConstructLap3D(const
00085 mtk::UniStgGrid3D &grid,
00086 int order_accuracy,
00087 mtk::Real mimetic_threshold) {
00088 mtk::Grad3D gg;
00089 mtk::Div3D dd;
00090
00091 bool info{gg.ConstructGrad3D(grid, order_accuracy, mimetic_threshold)};
00092
00093 #ifdef MTK_PERFORM_PREVENTIONS
00094 if (!info) {
00095 std::cerr << "Mimetic lap could not be built." << std::endl;
00096 return info;
00097 }
00098 #endif
00099
00100 info = dd.ConstructDiv3D(grid, order_accuracy, mimetic_threshold);
00101
00102 #ifdef MTK_PERFORM_PREVENTIONS
00103 if (!info) {
00104 std::cerr << "Mimetic div could not be built." << std::endl;
00105 return info;
00106 }
00107 #endif
00108
00109 mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00110 mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00111
00112 laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
00113
00114 return info;
00115 }
00116
00117 mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix() const {
00118 return laplacian_;
00119 }
00120
00121
00122 mtk::Real *mtk::Lap3D::data() const {
00123
00124 return laplacian_.data();
00125 }

```

## 18.105 src/mtk\_lapack\_adapter.cc File Reference

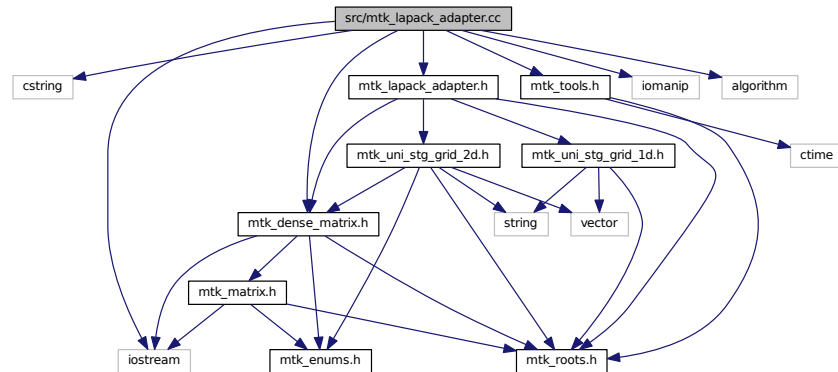
Adapter class for the LAPACK API.

```

#include <cstring>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_dense_matrix.h"
#include "mtk_lapack_adapter.h"

```

Include dependency graph for mtk\_lapack\_adapter.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- void [mtk::sgesv\\_](#) (int \*n, int \*nrhs, Real \*a, int \*lda, int \*ipiv, Real \*b, int \*ldb, int \*info)
- void [mtk::sgels\\_](#) (char \*trans, int \*m, int \*n, int \*nrhs, Real \*a, int \*lda, Real \*b, int \*ldb, Real \*work, int \*lwork, int \*info)  
*Single-precision GEneral matrix Least Squares solver.*
- void [mtk::sgeqrf\\_](#) (int \*m, int \*n, Real \*a, int \*lda, Real \*tau, Real \*work, int \*lwork, int \*info)  
*Single-precision GEneral matrix QR Factorization.*
- void [mtk::sormqr\\_](#) (char \*side, char \*trans, int \*m, int \*n, int \*k, Real \*a, int \*lda, Real \*tau, Real \*c, int \*ldc, Real \*work, int \*lwork, int \*info)  
*Single-precision Orthogonal [Matrix](#) from QR factorization.*

### 18.105.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the LAPACK.

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

**Todo** Write documentation using LaTeX.

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lapack\\_adapter.cc](#).

**18.106 mtk\_lapack\_adapter.cc**

```

00001
00022 /*
00023 Copyright (C) 2015, Computational Science Research Center, San Diego State
00024 University. All rights reserved.
00025
00026 Redistribution and use in source and binary forms, with or without modification,
00027 are permitted provided that the following conditions are met:
00028
00029 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00030 and a copy of the modified files should be reported once modifications are
00031 completed, unless these modifications are made through the project's GitHub
00032 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00033 should be developed and included in any deliverable.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions in binary form must reproduce the above copyright notice,
00039 this list of conditions and the following disclaimer in the documentation and/or
00040 other materials provided with the distribution.
00041
00042 4. Usage of the binary form on proprietary applications shall require explicit
00043 prior written permission from the the copyright holders, and due credit should
00044 be given to the copyright holders.
00045
00046 5. Neither the name of the copyright holder nor the names of its contributors
00047 may be used to endorse or promote products derived from this software without
00048 specific prior written permission.
00049
00050 The copyright holders provide no reassurances that the source code provided does
00051 not infringe any patent, copyright, or any other intellectual property rights of
00052 third parties. The copyright holders disclaim any liability to any recipient for
00053 claims brought against recipient by any third party for infringement of that
00054 parties intellectual property rights.
00055
00056 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00057 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00058 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00059 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00060 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00061 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00062 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00063 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00064 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00065 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00066 */
00067
00068 #include <cstring>
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <algorithm>
00074
00075 #include "mtk_tools.h"
00076 #include "mtk_dense_matrix.h"
00077 #include "mtk_lapack_adapter.h"
00078
00079 namespace mtk {
00080
00081 extern "C" {
00082
00083 #ifdef MTK_PRECISION_DOUBLE
00084
00103 void dgesv_(int* n,
00104 int* nrhs,
00105 Real* a,
00106 int* lda,

```

```
00107 int* ipiv,
00108 Real* b,
00109 int* ldb,
00110 int* info);
00111 #else
00112
00131 void sgesv_(int* n,
00132 int* nrhs,
00133 Real* a,
00134 int* lda,
00135 int* ipiv,
00136 Real* b,
00137 int* ldb,
00138 int* info);
00139 #endif
00140
00141 #ifdef MTK_PRECISION_DOUBLE
00142
00185 void dgels_(char* trans,
00186 int* m,
00187 int* n,
00188 int* nrhs,
00189 Real* a,
00190 int* lda,
00191 Real* b,
00192 int* ldb,
00193 Real* work,
00194 int* lwork,
00195 int* info);
00196 #else
00197
00240 void sgels_(char* trans,
00241 int* m,
00242 int* n,
00243 int* nrhs,
00244 Real* a,
00245 int* lda,
00246 Real* b,
00247 int* ldb,
00248 Real* work,
00249 int* lwork,
00250 int* info);
00251 #endif
00252
00253 #ifdef MTK_PRECISION_DOUBLE
00254
00283 void dgeqrf_(int *m,
00284 int *n,
00285 Real *a,
00286 int *lda,
00287 Real *tau,
00288 Real *work,
00289 int *lwork,
00290 int *info);
00291 #else
00292
00321 void sgeqrf_(int *m,
00322 int *n,
00323 Real *a,
00324 int *lda,
00325 Real *tau,
00326 Real *work,
00327 int *lwork,
00328 int *info);
00329 #endif
00330
00331 #ifdef MTK_PRECISION_DOUBLE
00332
00366 void dormqr_(char *side,
00367 char *trans,
00368 int *m,
00369 int *n,
00370 int *k,
00371 Real *a,
00372 int *lda,
00373 Real *tau,
00374 Real *c,
00375 int *ldc,
00376 Real *work,
00377 int *lwork,
00378 int *info);
```

```

00379 #else
00380
00414 void sormqr_(char *side,
00415 char *trans,
00416 int *m,
00417 int *n,
00418 int *k,
00419 Real *a,
00420 int *lda,
00421 Real *tau,
00422 Real *c,
00423 int *ldc,
00424 Real *work,
00425 int *lwork,
00426 int *info);
00427 #endif
00428 }
00429 }
00430
00431 int mtk::LAPACKAdapter::SolveDenseSystem(
 mtk::DenseMatrix &mm,
 mtk::Real *rhs) {
00432
00433 #ifdef MTK_PERFORM_PREVENTIONS
00434 mtk::Tools::Prevent(rhs == nullptr, __FILE__, __LINE__, __func__);
00435 #endif
00436
00437 int *ipiv{}; // Array for pivoting information.
00438 int nrhs{1}; // Number of right-hand sides.
00439 int info{}; // Status of the solution.
00440 int mm_rank{mm.num_rows()}; // Rank of the matrix.
00441
00442 try {
00443 ipiv = new int[mm_rank];
00444 } catch (std::bad_alloc &memory_allocation_exception) {
00445 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00446 std::endl;
00447 std::cerr << memory_allocation_exception.what() << std::endl;
00448 }
00449 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00450
00451 int ldbb = mm_rank;
00452 int mm_ld = mm_rank;
00453
00454 #ifdef MTK_PRECISION_DOUBLE
00455 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00456 #else
00457 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00458 #endif
00459
00460 delete [] ipiv;
00461
00462 return info;
00463 }
00464
00465 int mtk::LAPACKAdapter::SolveDenseSystem(
 mtk::DenseMatrix &mm,
 mtk::DenseMatrix &bb) {
00466
00467 int nrhs{bb.num_rows()}; // Number of right-hand sides.
00468
00469 #ifdef MTK_PERFORM_PREVENTIONS
00470 mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00471 #endif
00472
00473 int *ipiv{}; // Array for pivoting information.
00474 int info{}; // Status of the solution.
00475 int mm_rank{mm.num_rows()}; // Rank of the matrix.
00476
00477 try {
00478 ipiv = new int[mm_rank];
00479 } catch (std::bad_alloc &memory_allocation_exception) {
00480 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00481 std::endl;
00482 std::cerr << memory_allocation_exception.what() << std::endl;
00483 }
00484 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00485
00486 int ldbb = mm_rank;
00487 int mm_ld = mm_rank;
00488
00489
00490

```

```

00491 #ifdef MTK_PRECISION_DOUBLE
00492 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00493 #else
00494 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00495 #endif
00496
00497 delete [] ipiv;
00498
00499 // After output, the data in the matrix will be column-major ordered.
00500
00501 bb.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00502
00503 #if MTK_VERBOSE_LEVEL > 12
00504 std::cout << "bb_col_maj_ord =" << std::endl;
00505 std::cout << bb << std::endl;
00506 #endif
00507
00508 bb.OrderRowMajor();
00509
00510 #if MTK_VERBOSE_LEVEL > 12
00511 std::cout << "bb_row_maj_ord =" << std::endl;
00512 std::cout << bb << std::endl;
00513 #endif
00514
00515 return info;
00516 }
00517
00518 int mtk::LAPACKAdapter::SolveDenseSystem(
 mtk::DenseMatrix &mm,
 mtk::UniStgGrid1D &rhs) {
00519
00520 int nrhs{1}; // Number of right-hand sides.
00521
00522 int *ipiv{}; // Array for pivoting information.
00523 int info{}; // Status of the solution.
00524 int mm_rank{mm.num_rows()}; // Rank of the matrix.
00525
00526 try {
00527 ipiv = new int[mm_rank];
00528 } catch (std::bad_alloc &memory_allocation_exception) {
00529 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00530 std::endl;
00531 std::cerr << memory_allocation_exception.what() << std::endl;
00532 }
00533 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00534
00535 int lddb = mm_rank;
00536 int mm_ld = mm_rank;
00537
00538 mm.OrderColMajor();
00539
00540 #ifdef MTK_PRECISION_DOUBLE
00541 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00542 rhs.discrete_field(), &lddb, &info);
00543 #else
00544 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00545 rhs.discrete_field(), &lddb, &info);
00546 #endif
00547
00548 mm.OrderRowMajor();
00549
00550 delete [] ipiv;
00551
00552 return info;
00553 }
00554
00555 int mtk::LAPACKAdapter::SolveDenseSystem(
 mtk::DenseMatrix &mm,
 mtk::UniStgGrid2D &rhs) {
00556
00557 int nrhs{1}; // Number of right-hand sides.
00558
00559 int *ipiv{}; // Array for pivoting information.
00560 int info{}; // Status of the solution.
00561 int mm_rank{mm.num_rows()}; // Rank of the matrix.
00562
00563 try {
00564 ipiv = new int[mm_rank];
00565 } catch (std::bad_alloc &memory_allocation_exception) {
00566 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00567 std::endl;
00568 }

```

```

00570 std::cerr << memory_allocation_exception.what() << std::endl;
00571 }
00572 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00573
00574 int ldbb = mm_rank;
00575 int mm_ld = mm_rank;
00576
00577 mm.OrderColMajor();
00578
00579 #ifdef MTK_PRECISION_DOUBLE
00580 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00581 rhs.discrete_field(), &ldbb, &info);
00582 #else
00583 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00584 rhs.discrete_field(), &ldbb, &info);
00585 #endif
00586
00587 mm.OrderRowMajor();
00588
00589 delete [] ipiv;
00590
00591 return info;
00592 }
00593
00594 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
(mtk::DenseMatrix &aa) {
00595
00596 mtk::Real *work{}; // Working array.
00597 mtk::Real *tau{}; // Array for the Householder scalars.
00598
00599 // Prepare to factorize: allocate and inquire for the value of lwork.
00600 try {
00601 work = new mtk::Real[1];
00602 } catch (std::bad_alloc &memory_allocation_exception) {
00603 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00604 std::endl;
00605 std::cerr << memory_allocation_exception.what() << std::endl;
00606 }
00607 memset(work, mtk::kZero, sizeof(aa.data()[0])*1);
00608
00609 int lwork{-1};
00610 int info{};
00611
00612 int aa_num_cols = aa.num_cols();
00613 int aaT_num_rows = aa.num_cols();
00614 int aaT_num_cols = aa.num_rows();
00615
00616 #if MTK_VERBOSE_LEVEL > 12
00617 std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00618 std::cout << aa << std::endl;
00619 #endif
00620
00621 #ifdef MTK_PRECISION_DOUBLE
00622 dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00623 tau,
00624 work, &lwork, &info);
00625 #else
00626 fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00627 tau,
00628 work, &lwork, &info);
00629 #endif
00630
00631 if (info == 0) {
00632 lwork = (int) work[0];
00633 } else {
00634 std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00635 std::endl;
00636 std::cerr << "Exiting..." << std::endl;
00637 }
00638
00639 #if MTK_VERBOSE_LEVEL > 10
00640 std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00641 << std::endl;
00642 #endif
00643
00644 delete [] work;
00645 work = nullptr;
00646
00647 // Once we know lwork, we can actually invoke the factorization:
00648 try {
00649 work = new mtk::Real [lwork];

```



```

00650 } catch (std::bad_alloc &memory_allocation_exception) {
00651 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00652 std::endl;
00653 std::cerr << memory_allocation_exception.what() << std::endl;
00654 }
00655 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00656
00657 int ltau = std::min(aaT_num_rows, aaT_num_cols);
00658
00659 try {
00660 tau = new mtk::Real [ltau];
00661 } catch (std::bad_alloc &memory_allocation_exception) {
00662 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00663 std::endl;
00664 std::cerr << memory_allocation_exception.what() << std::endl;
00665 }
00666 memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00667
00668 #ifdef MTK_PRECISION_DOUBLE
00669 dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00670 tau, work, &lwork, &info);
00671 #else
00672 fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00673 tau, work, &lwork, &info);
00674 #endif
00675
00676 #ifdef MTK_PERFORM_PREVENTIONS
00677 if (!info) {
00678 std::cout << "QR factorization completed!" << std::endl << std::endl;
00679 } else {
00680 std::cerr << "Error solving system! info = " << info << std::endl;
00681 std::cerr << "Exiting..." << std::endl;
00682 }
00683 #endif
00684
00685 #if MTK_VERBOSE_LEVEL > 12
00686 std::cout << "Input matrix AFTER QR factorization:" << std::endl;
00687 std::cout << aa << std::endl;
00688 #endif
00689
00690 // We now generate the real matrix Q with orthonormal columns. This has to
00691 // be done separately since the actual output of dgeqrf_ (AA_) represents
00692 // the orthogonal matrix Q as a product of min(aa_num_rows, aa_num_cols)
00693 // elementary Householder reflectors. Notice that we must re-inquire the new
00694 // value for lwork that is used.
00695
00696 bool padded{false};
00697
00698 bool transpose{false};
00699
00700 mtk::DenseMatrix QQ_(aa.num_cols(), padded, transpose);
00701
00702 #if MTK_VERBOSE_LEVEL > 12
00703 std::cout << "Initialized QQ_T: " << std::endl;
00704 std::cout << QQ_ << std::endl;
00705 #endif
00706
00707 // Assemble the QQ_ matrix:
00708 lwork = -1;
00709
00710 delete[] work;
00711 work = nullptr;
00712
00713 try {
00714 work = new mtk::Real[l];
00715 } catch (std::bad_alloc &memory_allocation_exception) {
00716 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00717 std::endl;
00718 std::cerr << memory_allocation_exception.what() <<
00719 std::endl;
00720 }
00721 memset(work, mtk::kZero, sizeof(work[0])*l);
00722
00723 char side_{'L'};
00724 char trans_{'N'};
00725
00726 int aux = QQ_.num_rows();
00727
00728 #ifdef MTK_PRECISION_DOUBLE
00729 dormqr_(&side_, &trans_,
00730 &aa_num_cols, &aa_num_cols, <au, aa.data(), &aaT_num_rows, tau,

```

```

00731 QQ_.data(), &aux, work, &lwork, &info);
00732 #else
00733 formqr_(&side_, &trans_,
00734 &aa_num_cols, &aa_num_cols, <au, aa.data(), &aaT_num_rows, tau,
00735 QQ_.data(), &aux, work, &lwork, &info);
00736 #endif
00737
00738 if (info == 0) {
00739 lwork = (int) work[0];
00740 } else {
00741 std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00742 std::cerr << "Exiting..." << std::endl;
00743 }
00744
00745 #if MTK_VERBOSE_LEVEL > 10
00746 std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00747 std::endl << std::endl;
00748 #endif
00749
00750 delete[] work;
00751 work = nullptr;
00752
00753 try {
00754 work = new mtk::Real[lwork];
00755 } catch (std::bad_alloc &memory_allocation_exception) {
00756 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00757 std::endl;
00758 std::cerr << memory_allocation_exception.what() << std::endl;
00759 }
00760 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00761
00762 #ifdef MTK_PRECISION_DOUBLE
00763 dormqr_(&side_, &trans_,
00764 &aa_num_cols, &aa_num_cols, <au, aa.data(), &aaT_num_rows, tau,
00765 QQ_.data(), &aux, work, &lwork, &info);
00766 #else
00767 formqr_(&side_, &trans_,
00768 &aa_num_cols, &aa_num_cols, <au, aa.data(), &aaT_num_rows, tau,
00769 QQ_.data(), &aux, work, &lwork, &info);
00770 #endif
00771
00772 #ifdef MTK_PERFORM_PREVENTIONS
00773 if (!info) {
00774 std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00775 } else {
00776 std::cerr << "Something went wrong solving system! info = " << info <<
00777 std::endl;
00778 std::cerr << "Exiting..." << std::endl;
00779 }
00780 #endif
00781
00782 delete[] work;
00783 work = nullptr;
00784
00785 delete[] tau;
00786 tau = nullptr;
00787
00788 return QQ_;
00789 }
00790
00791 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
 mtk::DenseMatrix &aa,
00792
00793 mtk::Real *ob_,
00794 int ob_ld_) {
00795
00796 // We first invoke the solver to query for the value of lwork. For this,
00797 // we must at least allocate enough space to allow access to WORK(1), or
00798 // work[0]:
00799
00800 // If LWORK = -1, then a workspace query is assumed; the routine only
00801 // calculates the optimal size of the WORK array, returns this value as
00802 // the first entry of the WORK array, and no error message related to
00803 // LWORK is issued by XERBLA.
00804
00805 mtk::Real *work{}; // Work array.
00806
00807 try {
00808 work = new mtk::Real[1];
00809 } catch (std::bad_alloc &memory_allocation_exception) {
00810 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00811 std::endl;

```

```

00811 std::cerr << memory_allocation_exception.what() << std::endl;
00812 }
00813 memset(work, mtk::kZero, sizeof(work[0])*1);
00814
00815 char trans_{'N'};
00816 int nrhs_{1};
00817 int info{0};
00818 int lwork{-1};
00819
00820 int AA_num_rows_ = aa.num_cols();
00821 int AA_num_cols_ = aa.num_rows();
00822 int AA_ld_ = std::max(1, aa.num_cols());
00823
00824 #ifdef MTK_PRECISION_DOUBLE
00825 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00826 ob_, &ob_ld_,
00827 work, &lwork, &info);
00828 #else
00829 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00830 ob_, &ob_ld_,
00831 work, &lwork, &info);
00832 #endif
00833
00834 if (info == 0) {
00835 lwork = (int) work[0];
00836 } else {
00837 std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00838 std::endl;
00839 std::cerr << "Exiting..." << std::endl;
00840 return info;
00841 }
00842
00843 #if MTK_VERBOSE_LEVEL > 10
00844 std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00845 std::endl << std::endl;
00846 #endif
00847
00848 // We then use lwork's new value to create the work array:
00849 delete[] work;
00850 work = nullptr;
00851
00852 try {
00853 work = new mtk::Real[lwork];
00854 } catch (std::bad_alloc &memory_allocation_exception) {
00855 std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00856 std::cerr << memory_allocation_exception.what() << std::endl;
00857 }
00858 memset(work, 0.0, sizeof(work[0])*lwork);
00859
00860 // We now invoke the solver again:
00861 #ifdef MTK_PRECISION_DOUBLE
00862 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00863 ob_, &ob_ld_,
00864 work, &lwork, &info);
00865 #else
00866 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00867 ob_, &ob_ld_,
00868 work, &lwork, &info);
00869 #endif
00870
00871 delete [] work;
00872 work = nullptr;
00873
00874 return info;
00875 }

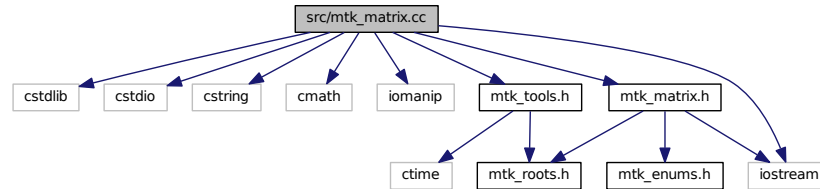
```

## 18.107 src/mtk\_matrix.cc File Reference

Implementing the representation of a matrix in the MTK.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <iostream>
#include "mtk_tools.h"
#include "mtk_matrix.h"
```

Include dependency graph for mtk\_matrix.cc:



### 18.107.1 Detailed Description

Implementation of the representation for the matrices implemented in the MTK.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_matrix.cc](#).

## 18.108 mtk\_matrix.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
```

```

00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <cstdlib>
00057 #include <cstdio>
00058 #include <cstring>
00059 #include <cmath>
00060
00061 #include <iomanip>
00062 #include <iostream>
00063
00064 #include "mtk_tools.h"
00065 #include "mtk_matrix.h"
00066
00067 mtk::Matrix::Matrix():
00068 storage_(mtk::MatrixStorage::DENSE),
00069 ordering_(mtk::MatrixOrdering::ROW_MAJOR),
00070 num_rows_(),
00071 num_cols_(),
00072 num_values_(),
00073 ld_(),
00074 num_zero_(),
00075 num_non_zero_(),
00076 num_null_(),
00077 num_non_null_(),
00078 kl_(),
00079 ku_(),
00080 bandwidth_(),
00081 abs_density_(),
00082 rel_density_(),
00083 abs_sparsity_(),
00084 rel_sparsity_() {}
00085
00086 mtk::Matrix::Matrix(const Matrix &in):
00087 storage_(in.storage_),
00088 ordering_(in.ordering_),
00089 num_rows_(in.num_rows_),
00090 num_cols_(in.num_cols_),
00091 num_values_(in.num_values_),
00092 ld_(in.ld_),
00093 num_zero_(in.num_zero_),
00094 num_non_zero_(in.num_non_zero_),
00095 num_null_(in.num_null_),
00096 num_non_null_(in.num_non_null_),
00097 kl_(in.kl_),
00098 ku_(in.ku_),
00099 bandwidth_(in.bandwidth_),
00100 abs_density_(in.abs_density_),
00101 rel_density_(in.rel_density_),
00102 abs_sparsity_(in.abs_sparsity_),
00103 rel_sparsity_(in.rel_sparsity_) {}
00104
00105 mtk::Matrix::~Matrix() noexcept {}
00106
00107 mtk::MatrixStorage mtk::Matrix::storage() const noexcept {
00108
00109 return storage_;
00110 }
00111
00112 mtk::MatrixOrdering mtk::Matrix::ordering() const noexcept {
00113
00114 return ordering_;
00115 }
00116
00117 int mtk::Matrix::num_rows() const noexcept {

```

```

00118
00119 return num_rows_;
00120 }
00121
00122 int mtk::Matrix::num_cols() const noexcept {
00123
00124 return num_cols_;
00125 }
00126
00127 int mtk::Matrix::num_values() const noexcept {
00128
00129 return num_values_;
00130 }
00131
00132 int mtk::Matrix::ld() const noexcept {
00133
00134 return ld_;
00135 }
00136
00137 int mtk::Matrix::num_zero() const noexcept {
00138
00139 return num_zero_;
00140 }
00141
00142 int mtk::Matrix::num_non_zero() const noexcept {
00143
00144 return num_non_zero_;
00145 }
00146
00147 int mtk::Matrix::num_null() const noexcept {
00148
00149 return num_null_;
00150 }
00151
00152 int mtk::Matrix::num_non_null() const noexcept {
00153
00154 return num_non_null_;
00155 }
00156
00157 int mtk::Matrix::kl() const noexcept {
00158
00159 return kl_;
00160 }
00161
00162 int mtk::Matrix::ku() const noexcept {
00163
00164 return ku_;
00165 }
00166
00167 int mtk::Matrix::bandwidth() const noexcept {
00168
00169 return bandwidth_;
00170 }
00171
00172 mtk::Real mtk::Matrix::rel_density() const noexcept {
00173
00174 return rel_density_;
00175 }
00176
00177 mtk::Real mtk::Matrix::abs_sparsity() const noexcept {
00178
00179 return abs_sparsity_;
00180 }
00181
00182 mtk::Real mtk::Matrix::rel_sparsity() const noexcept {
00183
00184 return rel_sparsity_;
00185 }
00186
00187 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss)
noexcept {
00188
00189 #ifdef MTK_PERFORM_PREVENTIONS
00190 mtk::Tools::Prevent(!(ss == mtk::MatrixStorage::DENSE ||
00191 ss == mtk::MatrixStorage::BANDED ||
00192 ss == mtk::MatrixStorage::CRS),
00193 __FILE__, __LINE__, __func__);
00194 #endif
00195
00196 storage_ = ss;
00197 }

```

```

00198
00199 void mtk::Matrix::set_ordering(const
 mtk::MatrixOrdering &oo) noexcept {
00200
00201 #ifdef MTK_PERFORM_PREVENTIONS
00202 bool aux{oo == mtk::MatrixOrdering::ROW_MAJOR ||
00203 oo == mtk::MatrixOrdering::COL_MAJOR};
00204 mtk::Tools::Prevent(!aux, __FILE__, __LINE__, __func__);
00205 #endif
00206
00207 ordering_ = oo;
00208
00209 ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00210 std::max(1,num_cols_): std::max(1,num_rows_);
00211 }
00212
00213 void mtk::Matrix::set_num_rows(const int &in) noexcept {
00214
00215 #ifdef MTK_PERFORM_PREVENTIONS
00216 mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00217 #endif
00218
00219 num_rows_ = in;
00220 num_values_ = num_rows_*num_cols_;
00221 ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00222 std::max(1,num_cols_): std::max(1,num_rows_);
00223 }
00224
00225 void mtk::Matrix::set_num_cols(const int &in) noexcept {
00226
00227 #ifdef MTK_PERFORM_PREVENTIONS
00228 mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00229 #endif
00230
00231 num_cols_ = in;
00232 num_values_ = num_rows_*num_cols_;
00233 ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00234 std::max(1,num_cols_): std::max(1,num_rows_);
00235 }
00236
00237 void mtk::Matrix::set_num_zero(const int &in) noexcept {
00238
00239 #ifdef MTK_PERFORM_PREVENTIONS
00240 mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00241 #endif
00242
00243 num_zero_ = in;
00244 num_non_zero_ = num_values_ - num_zero_;
00245
00246 rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00247 rel_sparsity_ = 1.0 - rel_density_;
00248 }
00249
00250
00251 void mtk::Matrix::set_num_null(const int &in) noexcept {
00252
00253 #ifdef MTK_PERFORM_PREVENTIONS
00254 mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00255 #endif
00256
00257 num_null_ = in;
00258 num_non_null_ = num_values_ - num_null_;
00259
00260 abs_density_ = (mtk::Real) num_non_null_/num_values_;
00261 abs_sparsity_ = 1.0 - abs_density_;
00262 }
00263
00264
00265 void mtk::Matrix::IncreaseNumZero() noexcept {
00266
00267 num_zero_++;
00268 num_non_zero_ = num_values_ - num_zero_;
00269 rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00270 rel_sparsity_ = 1.0 - rel_density_;
00271 }
00272
00273
00274
00275 void mtk::Matrix::IncreaseNumNull() noexcept {
00276
00277 num_null_++;
00278 num_non_null_ = num_values_ - num_null_;
00279 abs_density_ = (mtk::Real) num_non_null_/num_values_;
00280 }
00281

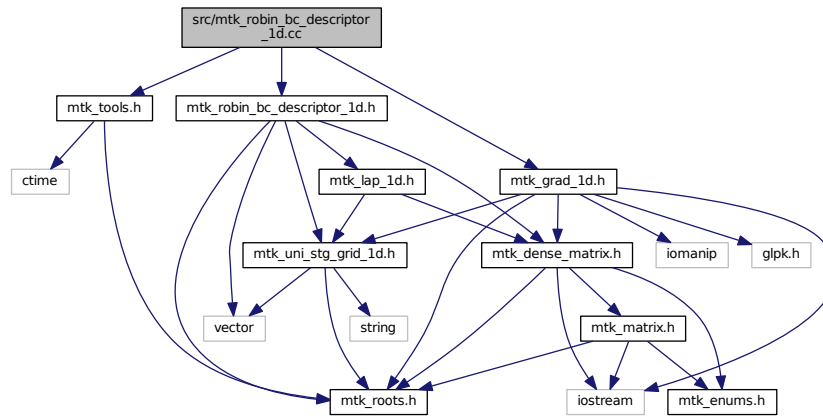
```

```
00282 abs_sparsity_ = 1.0 - abs_density_;
00283 }
```

## 18.109 src/mtk\_robin\_bc\_descriptor\_1d.cc File Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include "mtk_tools.h"
#include "mtk_grad_1d.h"
#include "mtk_robin_bc_descriptor_1d.h"
Include dependency graph for mtk_robin_bc_descriptor_1d.cc:
```



### 18.109.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 1D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ( $\partial\Omega = \{a, b\} \subset \mathbb{R}$ ), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) = \beta_b(b, t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.



## See also

<http://mathworld.wolfram.com/NormalVector.html>

## Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_1d.cc](#).

## 18.110 mtk\_robin\_bc\_descriptor\_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include "mtk_tools.h"
00090 #include "mtk_grad_1d.h"
00091 #include "mtk_robin_bc_descriptor_1d.h"
00092
00093 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D():
00094 highest_order_diff_west_(-1),
00095 highest_order_diff_east_(-1),
00096 west_condition_(nullptr),
00097 east_condition_(nullptr) {}
00098
00099 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D(
00100 const mtk::RobinBCDescriptor1D &desc):
00101 highest_order_diff_west_(desc.highest_order_diff_west_),
00102 highest_order_diff_east_(desc.highest_order_diff_east_),

```

```

00103 west_condition_(desc.west_condition_),
00104 east_condition_(desc.east_condition_) {}
00105
00106 mtk::RobinBCDescriptor1D::~RobinBCDescriptor1D() noexcept {}
00107
00108 int mtk::RobinBCDescriptor1D::highest_order_diff_west()
00109 const noexcept {
00110 return highest_order_diff_west_;
00111 }
00112
00113 int mtk::RobinBCDescriptor1D::highest_order_diff_east()
00114 const noexcept {
00115 return highest_order_diff_east_;
00116 }
00117
00118 void mtk::RobinBCDescriptor1D::PushBackWestCoeff(
00119 mtk::CoefficientFunction0D cw) {
00120
00121 #ifdef MTK_PERFORM_PREVENTIONS
00122 mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00123 mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00124 __FILE__, __LINE__, __func__);
00125 #endif
00126
00127 west_coefficients_.push_back(cw);
00128
00129 highest_order_diff_west_++;
00130 }
00131
00132 void mtk::RobinBCDescriptor1D::PushBackEastCoeff(
00133 mtk::CoefficientFunction0D ce) {
00134
00135 #ifdef MTK_PERFORM_PREVENTIONS
00136 mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00137 mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00138 __FILE__, __LINE__, __func__);
00139 #endif
00140
00141 east_coefficients_.push_back(ce);
00142
00143 highest_order_diff_east_++;
00144 }
00145
00146 void mtk::RobinBCDescriptor1D::set_west_condition(
00147 mtk::Real (*west_condition)(const mtk::Real &tt)) noexcept {
00148
00149 #ifdef MTK_PERFORM_PREVENTIONS
00150 mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00151 #endif
00152
00153 west_condition_ = west_condition;
00154 }
00155
00156 void mtk::RobinBCDescriptor1D::set_east_condition(
00157 mtk::Real (*east_condition)(const mtk::Real &tt)) noexcept {
00158
00159 #ifdef MTK_PERFORM_PREVENTIONS
00160 mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00161 #endif
00162
00163 east_condition_ = east_condition;
00164 }
00165
00166 bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix(
00167 const mtk::LaplD &lap,
00168 mtk::DenseMatrix &matrix,
00169 const mtk::Real &time) const {
00170
00171 #ifdef MTK_PERFORM_PREVENTIONS
00172 mtk::Tools::Prevent(highest_order_diff_west_ == -1,
00173 __FILE__, __LINE__, __func__);
00174 mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00175 __FILE__, __LINE__, __func__);
00176 mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00177 mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00178 #endif
00179
00180 matrix.SetValue(0, 0, (west_coefficients_[0])(time));
00183

```

```

00185 matrix.SetValue(matrix.num_rows() - 1,
00186 matrix.num_cols() - 1,
00187 (east_coefficients_[0])(time));
00188
00190 if (highest_order_diff_west_ > 0) {
00191
00193 mtk::Grad1D grad;
00194 if (!grad.ConstructGrad1D(lap.order_accuracy(),
00195 lap.mimetic_threshold())) {
00196 return false;
00197 }
00198
00200 mtk::DenseMatrix coeffs(grad.mim_bndy());
00201
00203 mtk::Real idx = mtk::kOne/lap.delta();
00204
00206 for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00207 mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00208 mtk::Real unit_normal{~mtk::kOne};
00209 aux *= unit_normal*(west_coefficients_[1])(time);
00210 matrix.SetValue(0, ii, matrix.GetValue(0, ii) + aux);
00211 }
00212
00214 for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00215 mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00216 mtk::Real unit_normal{mtk::kOne};
00217 aux *= -unit_normal*(east_coefficients_[1])(time);
00218 matrix.SetValue(matrix.num_rows() - 1,
00219 matrix.num_rows() - 1 - ii,
00220 matrix.GetValue(matrix.num_rows() - 1,
00221 matrix.num_rows() - 1 - ii) + aux);
00222 }
00223 }
00224 return true;
00225 }
00226
00228 void mtk::RobinBCDescriptor1D::ImposeOnGrid(
00229 UniStgGrid1D &grid,
00230 const mtk::Real &time) const {
00231
00232 #ifdef MTK_PERFORM_PREVENTIONS
00233 mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00234 mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00235 mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00236 #endif
00237
00238 (grid.discrete_field())[0] = west_condition_(time);
00239 (grid.discrete_field())[grid.num_cells_x() + 1] = east_condition_(time);
00240 }

```

## 18.111 src/mtk\_robin\_bc\_descriptor\_2d.cc File Reference

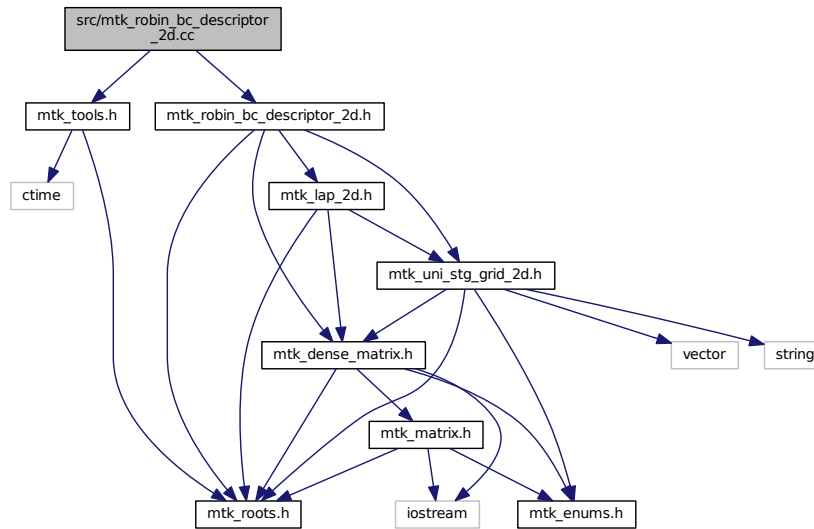
Impose Robin boundary conditions on the operators and on the grids.

```

#include "mtk_tools.h"
#include "mtk_robin_bc_descriptor_2d.h"

```

Include dependency graph for mtk\_robin\_bc\_descriptor\_2d.cc:



### 18.111.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 2D mimetic operators and the grids they are acting on.

**Def.** Let  $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  be the solution to an ordinary or partial differential equation of interest. We say that  $u$  satisfies a **Robin boundary condition** on  $\partial\Omega$  if and only if there exists  $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$  so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field  $u$  and its first normal derivative, in order for  $u$  to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_2d.cc](#).

## 18.112 mtk\_robin\_bc\_descriptor\_2d.cc

```

00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #include "mtk_tools.h"
00081
00082 #include "mtk_robin_bc_descriptor_2d.h"
00083
00084 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D() :
00085 highest_order_diff_west_(-1),
00086 highest_order_diff_east_(-1),
00087 highest_order_diff_south_(-1),
00088 highest_order_diff_north_(-1),
00089 west_condition_(),
00090 east_condition_(),
00091 south_condition_(),
00092 north_condition_() {}
00093
00094 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D(
00095 const mtk::RobinBCDescriptor2D &desc):
00096 highest_order_diff_west_(desc.highest_order_diff_west_),
00097 highest_order_diff_east_(desc.highest_order_diff_east_),
00098 highest_order_diff_south_(desc.highest_order_diff_south_),
00099 highest_order_diff_north_(desc.highest_order_diff_north_),
00100 west_condition_(desc.west_condition_),
00101 east_condition_(desc.east_condition_),
00102 south_condition_(desc.south_condition_),
00103 north_condition_(desc.north_condition_) {}
00104
00105 mtk::RobinBCDescriptor2D::~~RobinBCDescriptor2D() noexcept {}
00106
00107 int mtk::RobinBCDescriptor2D::highest_order_diff_west()
00108 const noexcept {
00109 return highest_order_diff_west_;

```

```

00110 }
00111
00112 int mtk::RobinBCDescriptor2D::highest_order_diff_east()
 const noexcept {
00113
00114 return highest_order_diff_east_;
00115 }
00116
00117 int mtk::RobinBCDescriptor2D::highest_order_diff_south()
 const noexcept {
00118
00119 return highest_order_diff_south_;
00120 }
00121
00122 int mtk::RobinBCDescriptor2D::highest_order_diff_north()
 const noexcept {
00123
00124 return highest_order_diff_north_;
00125 }
00126
00127 void mtk::RobinBCDescriptor2D::PushBackWestCoeff(
 mtk::CoefficientFunction1D cw) {
00128
00129 #ifdef MTK_PERFORM_PREVENTIONS
00130 mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00131 mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00132 __FILE__, __LINE__, __func__);
00133 #endif
00134
00135 west_coefficients_.push_back(cw);
00136
00137 highest_order_diff_west_++;
00138 }
00139
00140
00141 void mtk::RobinBCDescriptor2D::PushBackEastCoeff(
 mtk::CoefficientFunction1D ce) {
00142
00143 #ifdef MTK_PERFORM_PREVENTIONS
00144 mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00145 mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00146 __FILE__, __LINE__, __func__);
00147 #endif
00148
00149 east_coefficients_.push_back(ce);
00150
00151 highest_order_diff_east_++;
00152 }
00153
00154
00155 void mtk::RobinBCDescriptor2D::PushBackSouthCoeff(
 mtk::CoefficientFunction1D cs) {
00156
00157 #ifdef MTK_PERFORM_PREVENTIONS
00158 mtk::Tools::Prevent(cs == nullptr, __FILE__, __LINE__, __func__);
00159 mtk::Tools::Prevent(highest_order_diff_south_ > 1,
00160 __FILE__, __LINE__, __func__);
00161 #endif
00162
00163 south_coefficients_.push_back(cs);
00164
00165 highest_order_diff_south_++;
00166 }
00167
00168
00169 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff(
 mtk::CoefficientFunction1D cn) {
00170
00171 #ifdef MTK_PERFORM_PREVENTIONS
00172 mtk::Tools::Prevent(cn == nullptr, __FILE__, __LINE__, __func__);
00173 mtk::Tools::Prevent(highest_order_diff_north_ > 1,
00174 __FILE__, __LINE__, __func__);
00175 #endif
00176
00177 north_coefficients_.push_back(cn);
00178
00179 highest_order_diff_north_++;
00180 }
00181
00182
00183 void mtk::RobinBCDescriptor2D::set_west_condition(
 mtk::Real (*west_condition)(const mtk::Real &yy,
 const mtk::Real &tt)) noexcept {
00184
00185 #ifdef MTK_PERFORM_PREVENTIONS
00186
00187

```

```

00188 mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00189 #endif
00190
00191 west_condition_ = west_condition;
00192 }
00193
00194 void mtk::RobinBCDescriptor2D::set_east_condition(
00195 mtk::Real (*east_condition)(const mtk::Real &yy,
00196 const mtk::Real &tt)) noexcept {
00197
00198 #ifdef MTK_PERFORM_PREVENTIONS
00199 mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00200 #endif
00201
00202 east_condition_ = east_condition;
00203 }
00204
00205 void mtk::RobinBCDescriptor2D::set_south_condition(
00206 mtk::Real (*south_condition)(const mtk::Real &xx,
00207 const mtk::Real &tt)) noexcept {
00208
00209 #ifdef MTK_PERFORM_PREVENTIONS
00210 mtk::Tools::Prevent(south_condition == nullptr,
00211 __FILE__, __LINE__, __func__);
00212 #endif
00213
00214 south_condition_ = south_condition;
00215 }
00216
00217 void mtk::RobinBCDescriptor2D::set_north_condition(
00218 mtk::Real (*north_condition)(const mtk::Real &xx,
00219 const mtk::Real &tt)) noexcept {
00220
00221 #ifdef MTK_PERFORM_PREVENTIONS
00222 mtk::Tools::Prevent(north_condition == nullptr,
00223 __FILE__, __LINE__, __func__);
00224 #endif
00225
00226 north_condition_ = north_condition;
00227 }
00228
00229 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace
00230 (
00231 const mtk::Lap2D &lap,
00232 const mtk::UniStgGrid2D &grid,
00233 mtk::DenseMatrix &matrix,
00234 const mtk::Real &time) const {
00235
00236 // For the south-west corner:
00237 auto cc = (south_coefficients_[0])(grid.west_bndy(), time);
00238
00239 #if MTK_VERBOSE_LEVEL > 2
00240 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00241 matrix.num_cols() << " columns." << std::endl;
00242 std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00243 #endif
00244
00245 matrix.SetValue(0, 0, cc);
00246
00247 // Compute first centers per dimension.
00248 auto first_center_x = grid.west_bndy() + grid.delta_x()/
00249 mtk::kTwo;
00250
00251 // For each entry on the diagonal (south boundary):
00252 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00253 // Evaluate next set spatial coordinates to evaluate the coefficient.
00254 mtk::Real xx = first_center_x + ii*grid.delta_x();
00255 // Evaluate and assign the Dirichlet coefficient.
00256 cc = (south_coefficients_[0])(xx, time);
00257
00258 #if MTK_VERBOSE_LEVEL > 2
00259 std::cout << "Setting at " << ii + 1 << ' ' << ii + 1 << std::endl;
00260 #endif
00261
00262 matrix.SetValue(ii + 1, ii + 1, cc);
00263 }
00264
00265 // For the south-east corner:
00266 cc = (south_coefficients_[0])(grid.east_bndy(), time);
00267

```

```

00268 #if MTK_VERBOSE_LEVEL > 2
00269 std::cout << "Setting at " << grid.num_cells_x() + 1 << ' ' <<
00270 grid.num_cells_x() + 1 << std::endl;
00271 #endif
00272
00273 matrix.SetValue(grid.num_cells_x() + 1, grid.num_cells_x() + 1, cc);
00274
00275 if (highest_order_diff_south_ > 0) {
00276
00277 }
00280
00281 return true;
00282 }
00283
00284 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace
(
00285 const mtk::Lap2D &lap,
00286 const mtk::UniStgGrid2D &grid,
00287 mtk::DenseMatrix &matrix,
00288 const mtk::Real &time) const {
00289
00290
00291
00292
00293 // For each entry on the diagonal:
00294 for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00295 // Evaluate next set spatial coordinates to evaluate the coefficient.
00296 mtk::Real xx{(grid.discrete_domain_x())[ii]};
00297 // Evaluate and assign the Dirichlet coefficient.
00298 mtk::Real cc = (south_coefficients_[0])(xx, time);
00299 matrix.SetValue(ii, ii, cc);
00300 }
00301
00302 if (highest_order_diff_south_ > 0) {
00303
00304 }
00305
00306 return true;
00307 }
00308
00309 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace
(
00310 const mtk::Lap2D &lap,
00311 const mtk::UniStgGrid2D &grid,
00312 mtk::DenseMatrix &matrix,
00313 const mtk::Real &time) const {
00314
00315 int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00316
00317
00318 // For the north-west corner:
00319 mtk::Real cc =
00320 (north_coefficients_[0])(grid.west_bndy(), time);
00321
00322 #if MTK_VERBOSE_LEVEL > 2
00323 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00324 matrix.num_cols() << " columns." << std::endl;
00325 std::cout << "Setting at " << north_offset << ' ' << north_offset <<
00326 std::endl;
00327 #endif
00328
00329 matrix.SetValue(north_offset, north_offset, cc);
00330
00331 // Compute first centers per dimension.
00332 auto first_center_x = grid.west_bndy() + grid.delta_x()/
mtk::kTwo;
00333
00334 // For each entry on the diagonal (north boundary):
00335 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00336 // Evaluate next set spatial coordinates to evaluate the coefficient.
00337 mtk::Real xx = first_center_x + ii*grid.delta_x();
00338 // Evaluate and assign the Dirichlet coefficient.
00339 cc = (north_coefficients_[0])(xx, time);
00340
00341 #if MTK_VERBOSE_LEVEL > 2
00342 std::cout << "Setting at " << north_offset + ii + 1 << ' ' <<
00343 north_offset + ii + 1 << std::endl;
00344 #endif
00345
00346 matrix.SetValue(north_offset + ii + 1, north_offset + ii + 1, cc);
00347 }
00348
00349
00350
00351
00352

```



```

00353 // For the north-east corner:
00354 cc = (north_coefficients_[0])(grid.east_bndy(), time);
00355
00356 #if MTK_VERBOSE_LEVEL > 2
00357 std::cout << "Setting at " << north_offset + grid.num_cells_x() + 1 <<
00358 ' ' << north_offset + grid.num_cells_x() + 1 << std::endl;
00359 #endif
00360
00361 matrix.SetValue(north_offset + grid.num_cells_x() + 1,
00362 north_offset + grid.num_cells_x() + 1, cc);
00363
00364 if (highest_order_diff_north_ > 0) {
00365 }
00366
00367 return true;
00370 }
00371
00372 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace
00373 (
00374 const mtk::Lap2D &lap,
00375 const mtk::UniStgGrid2D &grid,
00376 mtk::DenseMatrix &matrix,
00377 const mtk::Real &time) const {
00378
00379 int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00380
00381 for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00382 mtk::Real xx{(grid.discrete_domain_x())[ii]};
00383 mtk::Real cc = (north_coefficients_[0])(xx, time);
00384 matrix.SetValue(north_offset + ii, north_offset + ii, cc);
00385 }
00386
00387 if (highest_order_diff_north_ > 0) {
00388 }
00389
00390 return true;
00391 }
00392
00393 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace
00394 (
00395 const mtk::Lap2D &lap,
00396 const mtk::UniStgGrid2D &grid,
00397 mtk::DenseMatrix &matrix,
00398 const mtk::Real &time) const {
00399
00400 // For the south-west corner:
00401 auto cc = (west_coefficients_[0])(grid.south_bndy(), time);
00402
00403 #if MTK_VERBOSE_LEVEL > 2
00404 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00405 matrix.num_cols() << " columns." << std::endl;
00406 std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00407 #endif
00408
00409 mtk::Real harmonic_mean = mtk::kOne/matrix.GetValue(0, 0) +
00410 mtk::kOne/cc;
00411 harmonic_mean = mtk::kTwo/harmonic_mean;
00412 matrix.SetValue(0, 0, harmonic_mean);
00413
00414 int west_offset{grid.num_cells_x() + 1};
00415
00416 auto first_center_y = grid.south_bndy() + grid.delta_y()/
00417 mtk::kTwo;
00418
00419 // For each west entry on the diagonal (west boundary):
00420 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00421 // Evaluate next set spatial coordinates to evaluate the coefficient.
00422 mtk::Real yy = first_center_y + ii*grid.delta_y();
00423 // Evaluate and assign the Dirichlet coefficient.
00424 cc = (west_coefficients_[0])(yy, time);
00425
00426 #if MTK_VERBOSE_LEVEL > 2
00427 std::cout << "Setting at " << west_offset + ii + 1 << ' ' <<
00428 west_offset + ii + 1 << std::endl;
00429 #endif
00430 }

```

```

00440
00441 matrix.SetValue(west_offset + ii + 1, west_offset + ii + 1, cc);
00442
00443 west_offset += grid.num_cells_x() + 1;
00444 }
00445
00446 // For the north-west corner:
00447 cc = (west_coefficients_[0])(grid.north_bndy(), time);
00448
00449 west_offset += grid.num_cells_x() + 1;
00450 int aux{west_offset};
00451 #if MTK_VERBOSE_LEVEL > 2
00452 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00453 #endif
00454
00455 harmonic_mean = mtk::kOne/matrix.GetValue(aux, aux) +
mtk::kOne/cc;
00456 harmonic_mean = mtk::kTwo/harmonic_mean;
00457
00458 matrix.SetValue(aux, aux, harmonic_mean);
00459
00460 if (highest_order_diff_west_ > 0) {
00461
00462 }
00463
00464 return true;
00465 }
00466 }
00467
00468 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace
(
00469 const mtk::Lap2D &lap,
00470 const mtk::UniStgGrid2D &grid,
00471 mtk::DenseMatrix &matrix,
00472 const mtk::Real &time) const {
00473
00474
00475
00476 int west_offset{grid.num_cells_x() + 1};
00477 // For each west entry on the diagonal:
00478 for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00479 // Evaluate next set spatial coordinates to evaluate the coefficient.
00480 mtk::Real yy{(grid.discrete_domain_y())[ii]};
00481 // Evaluate and assign the Dirichlet coefficient.
00482 mtk::Real cc = (west_coefficients_[0])(yy, time);
00483 matrix.SetValue(west_offset + ii, west_offset + ii, cc);
00484 west_offset += grid.num_cells_x() + 1;
00485 }
00486
00487 if (highest_order_diff_west_ > 0) {
00488
00489 }
00490
00491
00492 return true;
00493 }
00494
00495 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace
(
00496 const mtk::Lap2D &lap,
00497 const mtk::UniStgGrid2D &grid,
00498 mtk::DenseMatrix &matrix,
00499 const mtk::Real &time) const {
00500
00501
00502
00503 // For the south-east corner:
00504 auto cc = (east_coefficients_[0])(grid.south_bndy(), time);
00505
00506 int east_offset{grid.num_cells_x() + 1};
00507 #if MTK_VERBOSE_LEVEL > 2
00508 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00509 matrix.num_cols() << " columns." << std::endl;
00510 std::cout << "Setting at " << east_offset << ' ' << east_offset <<
00511 std::endl;
00512 #endif
00513
00514 mtk::Real harmonic_mean =
00515 mtk::kOne/matrix.GetValue(east_offset, east_offset) +
mtk::kOne/cc;
00516 harmonic_mean = mtk::kTwo/harmonic_mean;
00517
00518 matrix.SetValue(east_offset, east_offset, harmonic_mean);
00519
00520 auto first_center_y = grid.south_bndy() + grid.delta_y()/

```

```

mtk::kTwo;
00521
00522 // For each east entry on the diagonal (east boundary):
00523 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00524
00525 east_offset += grid.num_cells_x() + 1;
00526
00527 // Evaluate next set spatial coordinates to evaluate the coefficient.
00528 mtk::Real yy = first_center_y + ii*grid.delta_y();
00529 // Evaluate and assign the Dirichlet coefficient.
00530 cc = (east_coefficients_[0])(yy, time);
00531
00532 #if MTK_VERBOSE_LEVEL > 2
00533 std::cout << "Setting at " << east_offset + ii + 1 << ' ' <<
00534 east_offset + ii + 1 << std::endl;
00535 #endif
00536
00537 matrix.SetValue(east_offset + ii + 1, east_offset + ii + 1, cc);
00538 }
00539
00540 // For the north-east corner:
00541 cc = (east_coefficients_[0])(grid.north_bndy(), time);
00542
00543 east_offset += grid.num_cells_x() + 1;
00544 east_offset += grid.num_cells_x() + 1;
00545 int aux{east_offset};
00546 #if MTK_VERBOSE_LEVEL > 2
00547 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00548 #endif
00549
00550 harmonic_mean =
00551 mtk::kOne/matrix.GetValue(aux, aux) + mtk::kOne/cc;
00552 harmonic_mean = mtk::kTwo/harmonic_mean;
00553
00554 matrix.SetValue(aux, aux, harmonic_mean);
00555
00556 if (highest_order_diff_east_ > 0) {
00557
00558 }
00559
00560 return true;
00561 }
00562 }
00563
00564 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace
(
00565 const mtk::Lap2D &lap,
00566 const mtk::UniStgGrid2D &grid,
00567 mtk::DenseMatrix &matrix,
00568 const mtk::Real &time) const {
00569
00570
00571
00572 int east_offset{grid.num_cells_x() + 1};
00573 // For each west entry on the diagonal:
00574 for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00575 east_offset += grid.num_cells_x() + 1;
00576 // Evaluate next set spatial coordinates to evaluate the coefficient.
00577 mtk::Real yy{(grid.discrete_domain_y())[ii]};
00578 // Evaluate and assign the arithmetic mean of Dirichlet coefficients.
00579 mtk::Real cc = (east_coefficients_[0])(yy, time);
00580 matrix.SetValue(east_offset + ii, east_offset + ii, cc);
00581 }
00582
00583 if (highest_order_diff_east_ > 0) {
00584
00585 }
00586
00587 return true;
00588 }
00589 }
00590
00591 bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix(
00592 const mtk::Lap2D &lap,
00593 const mtk::UniStgGrid2D &grid,
00594 mtk::DenseMatrix &matrix,
00595 const mtk::Real &time) const {
00596
00597 #ifdef MTK_PERFORM_PREVENTIONS
00598 mtk::Tools::Prevent(highest_order_diff_south_ == -1,
00599 __FILE__, __LINE__, __func__);
00600 mtk::Tools::Prevent(highest_order_diff_north_ == -1,
00601 __FILE__, __LINE__, __func__);
00602 mtk::Tools::Prevent(highest_order_diff_west_ == -1,

```

```

00603 __FILE__, __LINE__, __func__);
00604 mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00605 __FILE__, __LINE__, __func__);
00606 mtk::Tools::Prevent(grid.nature() != mtk::SCALAR,
00607 __FILE__, __LINE__, __func__);
00608 mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00609 mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00610 mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00611 mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00612 #endif
00613
00616
00617 bool success{true};
00618
00619 if (!grid.Bound()) {
00620 success = ImposeOnSouthBoundaryNoSpace(lap, grid, matrix, time);
00621 #ifdef MTK_PERFORM_PREVENTIONS
00622 if (!success) {
00623 return false;
00624 }
00625 #endif
00626 success = ImposeOnNorthBoundaryNoSpace(lap, grid, matrix, time);
00627 #ifdef MTK_PERFORM_PREVENTIONS
00628 if (!success) {
00629 return false;
00630 }
00631 #endif
00632 success = ImposeOnWestBoundaryNoSpace(lap, grid, matrix, time);
00633 #ifdef MTK_PERFORM_PREVENTIONS
00634 if (!success) {
00635 return false;
00636 }
00637 #endif
00638 success = ImposeOnEastBoundaryNoSpace(lap, grid, matrix, time);
00639 #ifdef MTK_PERFORM_PREVENTIONS
00640 if (!success) {
00641 return false;
00642 }
00643 #endif
00644 } else {
00645 success = ImposeOnSouthBoundaryWithSpace(lap, grid, matrix, time);
00646 #ifdef MTK_PERFORM_PREVENTIONS
00647 if (!success) {
00648 return false;
00649 }
00650 #endif
00651 success = ImposeOnNorthBoundaryWithSpace(lap, grid, matrix, time);
00652 #ifdef MTK_PERFORM_PREVENTIONS
00653 if (!success) {
00654 return false;
00655 }
00656 #endif
00657 success = ImposeOnWestBoundaryWithSpace(lap, grid, matrix, time);
00658 #ifdef MTK_PERFORM_PREVENTIONS
00659 if (!success) {
00660 return false;
00661 }
00662 #endif
00663 success = ImposeOnEastBoundaryWithSpace(lap, grid, matrix, time);
00664 #ifdef MTK_PERFORM_PREVENTIONS
00665 if (!success) {
00666 return false;
00667 }
00668 #endif
00669 }
00670
00671 return success;
00672 }
00673
00674 void mtk::RobinBCDescriptor2D::ImposeOnGrid(
00675 mtk::UniStgGrid2D &grid,
00676 const mtk::Real &time) const {
00677
00678 #ifdef MTK_PERFORM_PREVENTIONS
00679 mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00680 mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00681 mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00682 mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00683 mtk::Tools::Prevent(south_condition_ == nullptr,
00684 __FILE__, __LINE__, __func__);
00685 mtk::Tools::Prevent(north_condition_ == nullptr,

```

```

00686 __FILE__, __LINE__, __func__);
00687 #endif
00688
00690 if (grid.nature() == mtk::SCALAR) {
00691
00693
00695 mtk::Real xx = grid.west_bndy();
00696 (grid.discrete_field())[0] = south_condition_(xx, time);
00697
00699 xx = xx + grid.delta_x()/mtk::kTwo;
00700 // For every point on the south boundary:
00701 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00702 (grid.discrete_field())[ii + 1] =
00703 south_condition_(xx + ii*grid.delta_x(), time);
00704 }
00705
00707 xx = grid.east_bndy();
00708 (grid.discrete_field())[grid.num_cells_x() + 1] =
00709 south_condition_(xx, time);
00710
00712
00714 xx = grid.west_bndy();
00715 int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00716 (grid.discrete_field())[north_offset] = north_condition_(xx, time);
00717
00719 xx = xx + grid.delta_x()/mtk::kTwo;
00720 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00721 (grid.discrete_field())[north_offset + ii + 1] =
00722 north_condition_(xx + ii*grid.delta_x(), time);
00723 }
00724
00726 xx = grid.east_bndy();
00727 (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00728 north_condition_(xx, time);
00729
00731
00733 mtk::Real yy = grid.south_bndy();
00734 (grid.discrete_field())[0] =
00735 ((grid.discrete_field())[0] + west_condition_(yy, time))/
00736 mtk::kTwo;
00737
00738 int west_offset{grid.num_cells_x() + 1 + 1};
00739 yy = yy + grid.delta_y()/mtk::kTwo;
00740 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00741 #if MTK_VERBOSE_LEVEL > 2
00742 std::cout << "Adding on " << west_offset << "-th position." << std::endl;
00743 #endif
00744 (grid.discrete_field())[west_offset] =
00745 west_condition_(yy + ii*grid.delta_y(), time);
00746 west_offset += grid.num_cells_x() + 1 + 1;
00747 }
00748
00750 yy = grid.north_bndy();
00751 north_offset = (grid.num_cells_y() + 1)*(grid.num_cells_x() + 2);
00752 (grid.discrete_field())[north_offset] =
00753 ((grid.discrete_field())[north_offset] + west_condition_(yy, time))/
00754 mtk::kTwo;
00755
00757 yy = grid.south_bndy();
00758 int east_offset{grid.num_cells_x() + 1};
00759 (grid.discrete_field())[east_offset] =
00760 ((grid.discrete_field())[east_offset] + east_condition_(yy, time))/
00761 mtk::kTwo;
00762
00764 yy = yy + grid.delta_y()/mtk::kTwo;
00765 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00766 east_offset += grid.num_cells_x() + 1 + 1;
00767 #if MTK_VERBOSE_LEVEL > 2
00768 std::cout << "Adding on " << east_offset << "-th position." << std::endl;
00769 #endif
00770 (grid.discrete_field())[east_offset] =
00771 east_condition_(yy + ii*grid.delta_y(), time);
00772 }
00773
00775 yy = grid.north_bndy();
00776 (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00777 ((grid.discrete_field())[north_offset + grid.num_cells_x() + 1] +
00778 east_condition_(yy, time))/mtk::kTwo;
00779
00781 } else {

```

```

00785
00787
00789 }
00790 }

```

### 18.113 src/mtk\_tools.cc File Reference

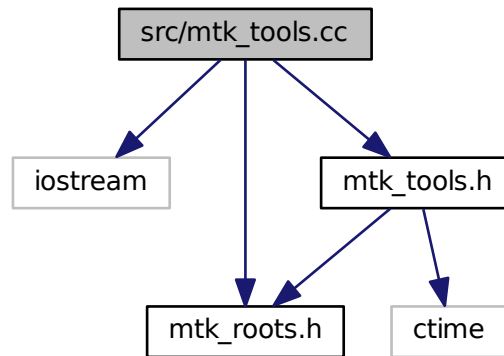
Tool manager class.

```

#include <iostream>
#include "mtk_roots.h"
#include "mtk_tools.h"

```

Include dependency graph for mtk\_tools.cc:



#### 18.113.1 Detailed Description

Implementation of a class providing basic tools to ensure execution correctness, and to assists with unitary testing.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_tools.cc](#).

### 18.114 mtk\_tools.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu

```

```

00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <iostream>
00058
00059 #include "mtk_roots.h"
00060 #include "mtk_tools.h"
00061
00062 void mtk::Tools::Prevent(const bool condition,
00063 const char *const fname,
00064 int lineno,
00065 const char *const fxname) noexcept {
00066
00067 if (lineno < 1) {
00068 std::cerr << __FILE__ << ": " << "Incorrect parameter at line " <<
00069 __LINE__ - 2 << " (" << __func__ << ")" << std::endl;
00070 exit(EXIT_FAILURE);
00071 }
00072
00073 if (condition) {
00074 std::cerr << fname << ": " << "Incorrect parameter at line " <<
00075 lineno << " (" << fxname << ")" << std::endl;
00076 exit(EXIT_FAILURE);
00077 }
00078 }
00079
00080
00081 int mtk::Tools::test_number_{}; // Current test being executed.
00082
00083 mtk::Real mtk::Tools::duration_{}; // Duration of the current test.
00084
00085 clock_t mtk::Tools::begin_time_{}; // Elapsed time on current test.
00086
00087 void mtk::Tools::BeginUnitTestNo(const int &nn) noexcept {
00088
00089 #if MTK_PERFORM_PREVENTIONS
00090 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00091 #endif
00092
00093 test_number_ = nn;
00094
00095 std::cout << "Beginning test " << nn << "." << std::endl;
00096 begin_time_ = clock();
00097 }
00098
00099 void mtk::Tools::EndUnitTestNo(const int &nn) noexcept {
00100

```

```

00101 #if MTK_PERFORM_PREVENTIONS
00102 mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00103 #endif
00104
00105 duration_ = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00106 }
00107
00108 void mtk::Tools::Assert(const bool &condition) noexcept {
00109
00110 if (condition) {
00111 std::cout << "Test " << test_number_ << ": PASSED in " << duration_ <<
00112 " s." << std::endl;
00113 } else {
00114 std::cout << "Test " << test_number_ << ": FAILED in " << duration_ <<
00115 " s." << std::endl;
00116 }
00117 }

```

## 18.115 src/mtk\_uni\_stg\_grid\_1d.cc File Reference

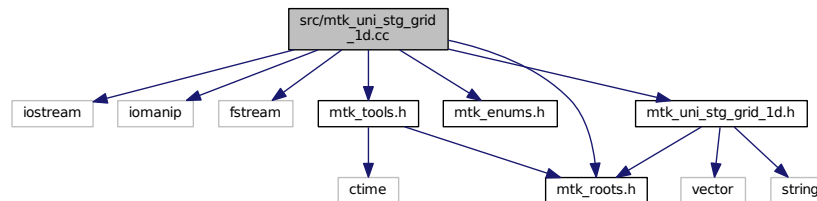
Implementation of an 1D uniform staggered grid.

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk\_uni\_stg\_grid\_1d.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::UniStgGrid1D &in)`

### 18.115.1 Detailed Description

Implementation of an 1D uniform staggered grid.



## Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

## 18.116 mtk\_uni\_stg\_grid\_1d.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_enums.h"
00062 #include "mtk_tools.h"
00063
00064 #include "mtk_uni_stg_grid_1d.h"
00065
00066 namespace mtk {
00067
00068 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid1D &in) {
00069
00070 stream << '[' << in.west_bndy_x << ':' << in.num_cells_x << ':' <<
00071 in.east_bndy_x << "]" = " << std::endl << std::endl;
00072
00073
00074
00075 stream << "x:";
00076 for (unsigned int ii = 0; ii < in.discrete_domain_x.size(); ++ii) {
00077 stream << std::setw(10) << in.discrete_domain_x[ii];

```

```

00078 }
00079 stream << std::endl;
00080
00082
00083 if (in.nature_ == mtk::SCALAR) {
00084 stream << "u:";
00085 }
00086 else {
00087 stream << "v:";
00088 }
00089 for (unsigned int ii = 0; ii < in.discrete_field_.size(); ++ii) {
00090 stream << std::setw(10) << in.discrete_field_[ii];
00091 }
00092
00093 stream << std::endl;
00094
00095 return stream;
00096 }
00097 }
00098
00099 mtk::UniStgGrid1D::UniStgGrid1D():
00100 nature_(),
00101 discrete_domain_x_(),
00102 discrete_field_(),
00103 west_bndy_x_(),
00104 east_bndy_x_(),
00105 num_cells_x_(),
00106 delta_x_() {}
00107
00108 mtk::UniStgGrid1D::UniStgGrid1D(const
UniStgGrid1D &grid):
00109 nature_(grid.nature_),
00110 west_bndy_x_(grid.west_bndy_x_),
00111 east_bndy_x_(grid.east_bndy_x_),
00112 num_cells_x_(grid.num_cells_x_),
00113 delta_x_(grid.delta_x_) {
00114
00115 std::copy(grid.discrete_domain_x_.begin(),
00116 grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
00117 discrete_domain_x_.begin());
00118
00119 std::copy(grid.discrete_field_.begin(),
00120 grid.discrete_field_.begin() + grid.discrete_field_.size(),
00121 discrete_field_.begin());
00122 }
00123
00124 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00125 const Real &east_bndy_x,
00126 const int &num_cells_x,
00127 const mtk::FieldNature &nature) {
00128
00129 #ifdef MTK_PERFORM_PREVENTIONS
00130 mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00131 mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00132 mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00133 mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00134 #endif
00135
00136 nature_ = nature;
00137 west_bndy_x_ = west_bndy_x;
00138 east_bndy_x_ = east_bndy_x;
00139 num_cells_x_ = num_cells_x;
00140
00141 delta_x_ = (east_bndy_x - west_bndy_x)/((mtk::Real) num_cells_x);
00142 }
00143
00144 mtk::UniStgGrid1D::~~UniStgGrid1D() {}
00145
00146 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00147
00148 return west_bndy_x_;
00149 }
00150
00151 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00152
00153 return east_bndy_x_;
00154 }
00155
00156 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00157

```

```

00158 return delta_x_;
00159 }
00160
00161 const mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() const
00162 {
00163 return discrete_domain_x_.data();
00164 }
00165
00166 mtk::Real *mtk::UniStgGrid1D::discrete_field() {
00167 return discrete_field_.data();
00168 }
00169 }
00170
00171 int mtk::UniStgGrid1D::num_cells_x() const {
00172 return num_cells_x_;
00173 }
00174 }
00175
00176 void mtk::UniStgGrid1D::BindScalarField(
00177 mtk::Real (*ScalarField)(const mtk::Real &xx)) {
00178
00179 #ifdef MTK_PERFORM_PREVENTIONS
00180 mtk::Tools::Prevent(nature_ == mtk::VECTOR, __FILE__, __LINE__, __func__);
00181 #endif
00182
00183 discrete_domain_x_.reserve(num_cells_x_ + 2);
00184
00185 discrete_domain_x_.push_back(west_bndy_x_);
00186 #ifdef MTK_PRECISION_DOUBLE
00187 auto first_center = west_bndy_x_ + delta_x_/2.0;
00188 #else
00189 auto first_center = west_bndy_x_ + delta_x_/2.0f;
00190 #endif
00191 discrete_domain_x_.push_back(first_center);
00192 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00193 discrete_domain_x_.push_back(first_center + ii*delta_x_);
00194 }
00195 discrete_domain_x_.push_back(east_bndy_x_);
00196
00197 discrete_field_.reserve(num_cells_x_ + 2);
00198 discrete_field_.push_back(ScalarField(west_bndy_x_));
00199 discrete_field_.push_back(ScalarField(first_center));
00200 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00201 discrete_field_.push_back(ScalarField(first_center + ii*delta_x_));
00202 }
00203 discrete_field_.push_back(ScalarField(east_bndy_x_));
00204 }
00205
00206 void mtk::UniStgGrid1D::BindVectorField(
00207 mtk::Real (*VectorField)(mtk::Real xx)) {
00208
00209 #ifdef MTK_PERFORM_PREVENTIONS
00210 mtk::Tools::Prevent(nature_ == mtk::SCALAR, __FILE__, __LINE__, __func__);
00211 #endif
00212
00213 discrete_domain_x_.reserve(num_cells_x_ + 1);
00214
00215 discrete_domain_x_.push_back(west_bndy_x_);
00216 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00217 discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00218 }
00219 discrete_domain_x_.push_back(east_bndy_x_);
00220
00221 discrete_field_.reserve(num_cells_x_ + 1);
00222 discrete_field_.push_back(VectorField(west_bndy_x_));
00223 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00224 discrete_field_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00225 }
00226 discrete_field_.push_back(VectorField(east_bndy_x_));
00227 }
00228
00229 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00230 std::string space_name,

```

```

00242 std::string field_name) const {
00243
00244 std::ofstream output_dat_file; // Output file.
00245 output_dat_file.open(filename);
00246
00247 if (!output_dat_file.is_open()) {
00248 return false;
00249 }
00250
00251 output_dat_file << "#" << space_name << " " << field_name << std::endl;
00252 for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00253 output_dat_file << discrete_domain_x_[ii] << " " << discrete_field_[ii] <<
00254 std::endl;
00255 }
00256
00257 output_dat_file.close();
00258
00259 return true;
00260 }
00261 }

```

## 18.117 src/mtk\_uni\_stg\_grid\_2d.cc File Reference

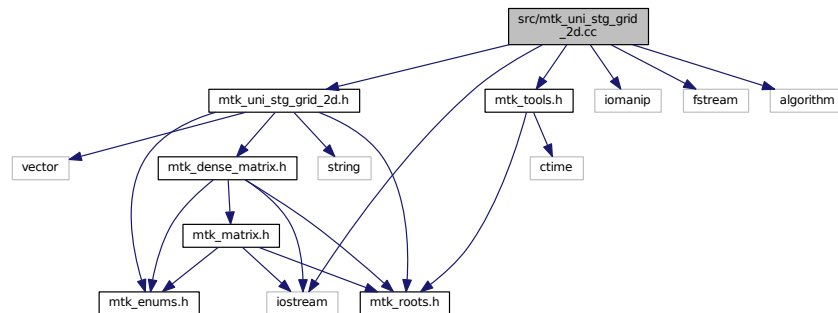
Implementation of a 2D uniform staggered grid.

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk\_uni\_stg\_grid\_2d.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::UniStgGrid2D &in)`

### 18.117.1 Detailed Description

Implementation of a 2D uniform staggered grid.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

## 18.118 mtk\_uni\_stg\_grid\_2d.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid2D &in) {
00068
00069 stream << '[' << in.west_bndy_ << ':' << in.num_cells_x << ':' <<

```

```

00070 in.east_bndy_ << "]" x ";
00071
00072 stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073 in.north_bndy_ << "]" = " << std::endl << std::endl;
00074
00075
00076
00077 stream << "x:";
00078 for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00079 stream << std::setw(10) << in.discrete_domain_x_[ii];
00080 }
00081 stream << std::endl;
00082
00083 stream << "y:";
00084 for (unsigned int ii = 0; ii < in.discrete_domain_y_.size(); ++ii) {
00085 stream << std::setw(10) << in.discrete_domain_y_[ii];
00086 }
00087 stream << std::endl;
00088
00089
00090
00091 if (in.nature_ == mtk::SCALAR) {
00092 stream << "u:" << std::endl;
00093 if (in.discrete_field_.size() > 0) {
00094 for (int ii = 0; ii < in.num_cells_x_ + 2; ++ii) {
00095 for (int jj = 0; jj < in.num_cells_y_ + 2; ++jj) {
00096 stream << std::setw(10) << in.discrete_field_[ii*in.
num_cells_y_ + jj];
00097 }
00098 stream << std::endl;
00099 }
00100 }
00101 } else {
00102
00103 int mm{in.num_cells_x_};
00104 int nn{in.num_cells_y_};
00105 int p_offset{nn*(mm + 1) - 1};
00106
00107 stream << "p(x,y):" << std::endl;
00108 for (int ii = 0; ii < nn; ++ii) {
00109 for (int jj = 0; jj < mm + 1; ++jj) {
00110 stream << std::setw(10) << in.discrete_field_[ii*(mm + 1) + jj];
00111 }
00112 stream << std::endl;
00113 }
00114 stream << std::endl;
00115
00116 stream << "q(x,y):" << std::endl;
00117 for (int ii = 0; ii < nn + 1; ++ii) {
00118 for (int jj = 0; jj < mm; ++jj) {
00119 stream << std::setw(10) <<
00120 in.discrete_field_[p_offset + ii*mm + jj];
00121 }
00122 stream << std::endl;
00123 }
00124 stream << std::endl;
00125 }
00126
00127 return stream;
00128 }
00129 }
00130
00131 mtk::UniStgGrid2D::UniStgGrid2D():
00132 discrete_domain_x_(),
00133 discrete_domain_y_(),
00134 discrete_field_(),
00135 nature_(),
00136 west_bndy_(),
00137 east_bndy_(),
00138 num_cells_x_(),
00139 delta_x_(),
00140 south_bndy_(),
00141 north_bndy_(),
00142 num_cells_y_(),
00143 delta_y_() {}
00144
00145 mtk::UniStgGrid2D::UniStgGrid2D(const
UniStgGrid2D &grid):
00146 nature_(grid.nature_),
00147 west_bndy_(grid.west_bndy_),
00148 east_bndy_(grid.east_bndy_),
00149 num_cells_x_(grid.num_cells_x_),
00150 delta_x_(grid.delta_x_),

```

```

00151 south_bndy_(grid.south_bndy_),
00152 north_bndy_(grid.north_bndy_),
00153 num_cells_y_(grid.num_cells_y_),
00154 delta_y_(grid.delta_y_) {
00155
00156 std::copy(grid.discrete_domain_x_.begin(),
00157 grid.discrete_domain_x_.begin() + grid.
00158 discrete_domain_x_.size(),
00159 discrete_domain_x_.begin());
00160 std::copy(grid.discrete_domain_y_.begin(),
00161 grid.discrete_domain_y_.begin() + grid.
00162 discrete_domain_y_.size(),
00163 discrete_domain_y_.begin());
00164 std::copy(grid.discrete_field_.begin(),
00165 grid.discrete_field_.begin() + grid.discrete_field_.size(),
00166 discrete_field_.begin());
00167 }
00168
00169 mtk::UniStgGrid2D::UniStgGrid2D(const Real &west_bndy,
00170 const Real &east_bndy,
00171 const int &num_cells_x,
00172 const Real &south_bndy,
00173 const Real &north_bndy,
00174 const int &num_cells_y,
00175 const mtk::FieldNature &nature) {
00176
00177 #ifdef MTK_PERFORM_PREVENTIONS
00178 mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00179 mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00180 mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00181 mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00182 mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00183 mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00184 mtk::Tools::Prevent(north_bndy <= south_bndy,
00185 __FILE__, __LINE__, __func__);
00186 mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00187 #endif
00188 nature_ = nature;
00189
00190 west_bndy_ = west_bndy;
00191 east_bndy_ = east_bndy;
00192 num_cells_x_ = num_cells_x;
00193
00194 south_bndy_ = south_bndy;
00195 north_bndy_ = north_bndy;
00196 num_cells_y_ = num_cells_y;
00197
00198 delta_x_ = (east_bndy_ - west_bndy_) / ((mtk::Real) num_cells_x);
00199 delta_y_ = (north_bndy_ - south_bndy_) / ((mtk::Real) num_cells_y);
00200 }
00201
00202
00203 mtk::UniStgGrid2D::~UniStgGrid2D() {}
00204
00205 mtk::FieldNature mtk::UniStgGrid2D::nature() const {
00206
00207 return nature_;
00208 }
00209
00210 mtk::Real mtk::UniStgGrid2D::west_bndy() const {
00211
00212 return west_bndy_;
00213 }
00214
00215 mtk::Real mtk::UniStgGrid2D::east_bndy() const {
00216
00217 return east_bndy_;
00218 }
00219
00220 int mtk::UniStgGrid2D::num_cells_x() const {
00221
00222 return num_cells_x_;
00223 }
00224
00225 mtk::Real mtk::UniStgGrid2D::delta_x() const {
00226
00227 return delta_x_;
00228 }
00229

```

```

00230 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_x() const
00231 {
00232 return discrete_domain_x_.data();
00233 }
00234
00235 mtk::Real mtk::UniStgGrid2D::south_bndy() const {
00236 return south_bndy_;
00237 }
00238
00239
00240 mtk::Real mtk::UniStgGrid2D::north_bndy() const {
00241 return north_bndy_;
00242 }
00243
00244
00245 int mtk::UniStgGrid2D::num_cells_y() const {
00246 return num_cells_y_;
00247 }
00248
00249
00250 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00251 return delta_y_;
00252 }
00253
00254
00255 bool mtk::UniStgGrid2D::Bound() const {
00256 return discrete_field_.size() != 0;
00257 }
00258
00259
00260 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_y() const
00261 {
00262 return discrete_domain_y_.data();
00263 }
00264
00265 mtk::Real* mtk::UniStgGrid2D::discrete_field() {
00266 return discrete_field_.data();
00267 }
00268
00269
00270 int mtk::UniStgGrid2D::Size() const {
00271 return discrete_field_.size();
00272 }
00273
00274
00275 void mtk::UniStgGrid2D::BindScalarField(
00276 Real (*ScalarField)(const Real &xx, const Real &yy)) {
00277
00278 #ifdef MTK_PERFORM_PREVENTIONS
00279 mtk::Tools::Prevent(nature_ != mtk::SCALAR, __FILE__, __LINE__, __func__);
00280 #endif
00281
00282
00283 discrete_domain_x_.reserve(num_cells_x_ + 2);
00284
00285 discrete_domain_x_.push_back(west_bndy_);
00286 #ifdef MTK_PRECISION_DOUBLE
00287 auto first_center = west_bndy_ + delta_x_/2.0;
00288 #else
00289 auto first_center = west_bndy_ + delta_x_/2.0f;
00290 #endif
00291 discrete_domain_x_.push_back(first_center);
00292 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00293 discrete_domain_x_.push_back(first_center + ii*delta_x_);
00294 }
00295 discrete_domain_x_.push_back(east_bndy_);
00296
00297
00298 discrete_domain_y_.reserve(num_cells_y_ + 2);
00299
00300 discrete_domain_y_.push_back(south_bndy_);
00301 #ifdef MTK_PRECISION_DOUBLE
00302 first_center = south_bndy_ + delta_x_/2.0;
00303 #else
00304 first_center = south_bndy_ + delta_x_/2.0f;
00305 #endif
00306 discrete_domain_y_.push_back(first_center);
00307 for (auto ii = 1; ii < num_cells_y_; ++ii) {
00308 discrete_domain_y_.push_back(first_center + ii*delta_y_);
00309 }
00310

```



```

00311 }
00312 discrete_domain_y_.push_back(north_bndy_);
00313
00315 discrete_field_.reserve((num_cells_x_ + 2)*(num_cells_y_ + 2));
00317
00318 for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00319 for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00320 #if MTK_VERBOSE_LEVEL > 6
00321 std::cout << "Pushing value for x = " << discrete_domain_x_[jj] <<
00322 " y = " << discrete_domain_y_[ii] << std::endl;
00323 #endif
00324 discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00325 discrete_domain_y_[ii]));
00326 }
00327 }
00328 }
00329
00330 void mtk::UniStgGrid2D::BindVectorFieldPComponent(
00331 mtk::Real (*VectorField)(const mtk::Real &xx, const
00332 mtk::Real &yy)) {
00333 int mm{num_cells_x_};
00334 int nn{num_cells_y_};
00335
00336 int total{nn*(mm + 1) + mm*(nn + 1)};
00337
00338 #ifdef MTK_PRECISION_DOUBLE
00339 double half_delta_x{delta_x_/2.0};
00340 double half_delta_y{delta_y_/2.0};
00341 #else
00342 float half_delta_x{delta_x_/2.0f};
00343 float half_delta_y{delta_y_/2.0f};
00344 #endif
00345
00347 // We need every data point of the discrete domain; i.e. we need all the
00348 // nodes and all the centers. There are mm centers for the x direction, and
00349 // nn centers for the y direction. Since there is one node per center, that
00350 // amounts to 2*mm. If we finally consider the final boundary node, it
00351 // amounts to a total of 2*mm + 1 for the x direction. Analogously, for the
00352 // y direction, this amounts to 2*nn + 1.
00353
00354 discrete_domain_x_.reserve(2*mm + 1);
00355
00356 discrete_domain_x_.push_back(west_bndy_);
00357 for (int ii = 1; ii < (2*mm + 1); ++ii) {
00358 discrete_domain_x_.push_back(west_bndy_ + ii*half_delta_x);
00359 }
00360
00361 discrete_domain_y_.reserve(2*nn + 1);
00362
00363 discrete_domain_y_.push_back(south_bndy_);
00364 for (int ii = 1; ii < (2*nn + 1); ++ii) {
00365 discrete_domain_y_.push_back(south_bndy_ + ii*half_delta_y);
00366 }
00367
00368 discrete_field_.reserve(total);
00369
00370 // For each y-center.
00371 for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00372 // Bind all of the x-nodes for this y-center.
00373 for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00374 discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00375 discrete_domain_y_[ii]));
00376
00377 #if MTK_VERBOSE_LEVEL > 6
00378 std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00379 discrete_domain_y_[ii] << " = " <<
00380 VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00381 #endif
00382 }
00383 }
00384
00385 #if MTK_VERBOSE_LEVEL > 6
00386 std::cout << std::endl;
00387 #endif
00388 }
00389
00390 #if MTK_VERBOSE_LEVEL > 6
00391 std::cout << std::endl;
00392 #endif
00393 }
00394

```

```

00395 void mtk::UniStgGrid2D::BindVectorFieldQComponent(
00396 mtk::Real (*VectorField)(const mtk::Real &xx, const
00397 mtk::Real &yy)) {
00398 int mm{num_cells_x_};
00399 int nn{num_cells_y_};
00400
00401 // For each y-node.
00402 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00403 // Bind all of the x-center for this y-node.
00404 for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00405 discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00406 discrete_domain_y_[ii]));
00407
00408 #if MTK_VERBOSE_LEVEL > 6
00409 std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00410 discrete_domain_y_[ii] << " = " <<
00411 VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00412 #endif
00413 }
00414 }
00415 #if MTK_VERBOSE_LEVEL > 6
00416 std::cout << std::endl;
00417 #endif
00418 }
00419 void mtk::UniStgGrid2D::BindVectorField(
00420 Real (*VectorFieldPComponent)(const Real &xx, const Real &yy),
00421 Real (*VectorFieldQComponent)(const Real &xx, const Real &yy)) {
00422 #ifdef MTK_PERFORM_PREVENTIONS
00423 mtk::Tools::Prevent(nature_ != mtk::VECTOR, __FILE__, __LINE__, __func__);
00424 #endif
00425 BindVectorFieldPComponent(VectorFieldPComponent);
00426 BindVectorFieldQComponent(VectorFieldQComponent);
00427 }
00428 bool mtk::UniStgGrid2D::WriteToFile(std::string filename,
00429 std::string space_name_x,
00430 std::string space_name_y,
00431 std::string field_name) const {
00432 std::ofstream output_dat_file; // Output file.
00433 output_dat_file.open(filename);
00434 if (!output_dat_file.is_open()) {
00435 return false;
00436 }
00437 if (nature_ == mtk::SCALAR) {
00438 output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00439 field_name << std::endl;
00440 int idx{};
00441 for (unsigned int ii = 0; ii < discrete_domain_y_.size(); ++ii) {
00442 for (unsigned int jj = 0; jj < discrete_domain_x_.size(); ++jj) {
00443 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00444 discrete_domain_y_[ii] << ' ' <<
00445 discrete_field_[idx] <<
00446 std::endl;
00447 idx++;
00448 }
00449 output_dat_file << std::endl;
00450 }
00451 } else {
00452 output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00453 field_name << std::endl;
00454 output_dat_file << "# Horizontal component:" << std::endl;
00455 int mm{num_cells_x_};
00456 int nn{num_cells_y_};
00457
00458 // For each y-center.
00459 int idx{};
00460 for (int ii = 1; ii < 2*nn + 1; ii += 2) {

```

```

00477 // Bind all of the x-nodes for this y-center.
00478 for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00479
00480 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00481 discrete_domain_y_[ii] << ' ' << discrete_field_[idx] << ' ' <<
00482 mtk::kZero << std::endl;
00483
00484 ++idx;
00485 }
00486 }
00487
00488 int p_offset{nn*(mm + 1) - 1};
00489 idx = 0;
00490 output_dat_file << "# Vertical component:" << std::endl;
00491 // For each y-node.
00492 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00493 // Bind all of the x-center for this y-node.
00494 for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00495
00496 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00497 discrete_domain_y_[ii] << ' ' << mtk::kZero << ' ' <<
00498 discrete_field_[p_offset + idx] << std::endl;
00499
00500 ++idx;
00501 }
00502 }
00503 }
00504 }
00505
00506 output_dat_file.close();
00507
00508 return true;
00509 }

```

## 18.119 src/mtk\_uni\_stg\_grid\_3d.cc File Reference

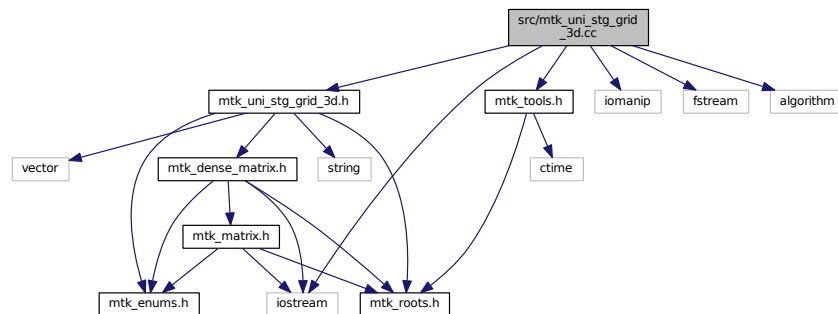
Implementation of a 2D uniform staggered grid.

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for mtk\_uni\_stg\_grid\_3d.cc:



## Namespaces

- [mtk](#)

*Mimetic Methods Toolkit namespace.*

## Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::UniStgGrid3D &in)`

### 18.119.1 Detailed Description

Implementation of a 2D uniform staggered grid.

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_3d.cc](#).

### 18.120 mtk\_uni\_stg\_grid\_3d.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>

```

```

00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid3D &in) {
00068
00069 stream << '[' << in.west_bndy_ << ':' << in.num_cells_x_ << ':' <<
00070 in.east_bndy_ << "]" x ";
00071
00072 stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073 in.north_bndy_ << "]" x ";
00074
00075 stream << '[' << in.bottom_bndy_ << ':' << in.num_cells_z_ << ':' <<
00076 in.top_bndy_ << "]" = " << std::endl << std::endl;
00077
00078 stream << "x:";
00079 for (auto const &cc: in.discrete_domain_x_) {
00080 stream << std::setw(10) << cc;
00081 }
00082 stream << std::endl;
00083
00084 stream << "y:";
00085 for (auto const &cc: in.discrete_domain_y_) {
00086 stream << std::setw(10) << cc;
00087 }
00088 stream << std::endl;
00089
00090 stream << "z:";
00091 for (auto const &cc: in.discrete_domain_z_) {
00092 stream << std::setw(10) << cc;
00093 }
00094 stream << std::endl;
00095
00096 if (in.nature_ == mtk::SCALAR) {
00097 stream << "u(x,y,z):" << std::endl;
00098 if (in.discrete_field_.size() > 0) {
00099
00100 }
00101 } else {
00102 stream << "p(x,y,z):" << std::endl;
00103 stream << "q(x,y,z):" << std::endl;
00104 if (in.discrete_field_.size() > 0) {
00105
00106 }
00107 }
00108 return stream;
00109 }
00110
00111 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator=(const
00112 mtk::UniStgGrid3D &in) {
00113
00114 UniStgGrid3D out(in);
00115
00116 return out;
00117 }
00118
00119 mtk::UniStgGrid3D::UniStgGrid3D():
00120 discrete_domain_x_(),
00121 discrete_domain_y_(),
00122 discrete_domain_z_(),
00123 discrete_field_(),
00124 nature_(),
00125 west_bndy_(),
00126 east_bndy_(),
00127 num_cells_x_(),
00128 delta_x_(),
00129 south_bndy_(),
00130 north_bndy_(),
00131 num_cells_y_(),
00132 delta_y_(),
00133 bottom_bndy_(),
00134 top_bndy_(),
00135 num_cells_z_(),
00136 delta_z_() {}

```

```

00141
00142 mtk::UniStgGrid3D::UniStgGrid3D(const
 UniStgGrid3D &grid):
00143 nature_(grid.nature_),
00144 west_bndy_(grid.west_bndy_),
00145 east_bndy_(grid.east_bndy_),
00146 num_cells_x_(grid.num_cells_x_),
00147 delta_x_(grid.delta_x_),
00148 south_bndy_(grid.south_bndy_),
00149 north_bndy_(grid.north_bndy_),
00150 num_cells_y_(grid.num_cells_y_),
00151 delta_y_(grid.delta_y_),
00152 bottom_bndy_(grid.bottom_bndy_),
00153 top_bndy_(grid.top_bndy_),
00154 num_cells_z_(grid.num_cells_z_),
00155 delta_z_(grid.delta_z_) {
00156
00157 std::copy(grid.discrete_domain_x_.begin(),
00158 grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
discrete_domain_x_.begin());
00159
00160
00161 std::copy(grid.discrete_domain_y_.begin(),
00162 grid.discrete_domain_y_.begin() + grid.
discrete_domain_y_.size(),
discrete_domain_y_.begin());
00163
00164
00165 std::copy(grid.discrete_domain_z_.begin(),
00166 grid.discrete_domain_z_.begin() + grid.
discrete_domain_z_.size(),
discrete_domain_z_.begin());
00167
00168
00169 std::copy(grid.discrete_field_.begin(),
00170 grid.discrete_field_.begin() + grid.discrete_field_.size(),
00171 discrete_field_.begin());
00172 }
00173
00174 mtk::UniStgGrid3D::UniStgGrid3D(const Real &west_bndy,
 const Real &east_bndy,
00175 const int &num_cells_x,
 const Real &south_bndy,
00176 const Real &north_bndy,
 const int &num_cells_y,
00177 const Real &bottom_bndy,
 const Real &top_bndy,
00178 const int &num_cells_z,
 const mtk::FieldNature &nature) {
00179
00180
00181
00182
00183
00184
00185 #ifdef MTK_PERFORM_PREVENTIONS
00186 mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00187 mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00188 mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00189 mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00190 mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00191 mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00192 mtk::Tools::Prevent(north_bndy <= south_bndy,
 __FILE__, __LINE__, __func__);
00193
00194 mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00195 mtk::Tools::Prevent(bottom_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00196 mtk::Tools::Prevent(top_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00197 mtk::Tools::Prevent(top_bndy <= bottom_bndy,
 __FILE__, __LINE__, __func__);
00198
00199 mtk::Tools::Prevent(num_cells_z < 0, __FILE__, __LINE__, __func__);
00200 #endif
00201
00202 nature_ = nature;
00203
00204 west_bndy_ = west_bndy;
00205 east_bndy_ = east_bndy;
00206 num_cells_x_ = num_cells_x;
00207
00208 south_bndy_ = south_bndy;
00209 north_bndy_ = north_bndy;
00210 num_cells_y_ = num_cells_y;
00211
00212 bottom_bndy_ = bottom_bndy;
00213 top_bndy_ = top_bndy;
00214 num_cells_z_ = num_cells_z;
00215
00216 delta_x_ = (east_bndy_ - west_bndy_) / ((mtk::Real) num_cells_x);
00217 delta_y_ = (north_bndy_ - south_bndy_) / ((mtk::Real) num_cells_y);

```

```

00218 delta_z_ = (top_bndy_ - bottom_bndy_) / ((mtk::Real) num_cells_z);
00219 }
00220
00221 mtk::UniStgGrid3D::~UniStgGrid3D() {}
00222
00223 mtk::FieldNature mtk::UniStgGrid3D::nature() const {
00224 return nature_;
00225 }
00226
00227 mtk::Real mtk::UniStgGrid3D::west_bndy() const {
00228 return west_bndy_;
00229 }
00230
00231 mtk::Real mtk::UniStgGrid3D::east_bndy() const {
00232 return east_bndy_;
00233 }
00234
00235 int mtk::UniStgGrid3D::num_cells_x() const {
00236 return num_cells_x_;
00237 }
00238
00239 mtk::Real mtk::UniStgGrid3D::delta_x() const {
00240 return delta_x_;
00241 }
00242
00243 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_x() const
00244 {
00245 return discrete_domain_x_.data();
00246 }
00247
00248 mtk::Real mtk::UniStgGrid3D::south_bndy() const {
00249 return south_bndy_;
00250 }
00251
00252 mtk::Real mtk::UniStgGrid3D::north_bndy() const {
00253 return north_bndy_;
00254 }
00255
00256 int mtk::UniStgGrid3D::num_cells_y() const {
00257 return num_cells_y_;
00258 }
00259
00260 mtk::Real mtk::UniStgGrid3D::delta_y() const {
00261 return delta_y_;
00262 }
00263
00264 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_y() const
00265 {
00266 return discrete_domain_y_.data();
00267 }
00268
00269 mtk::Real mtk::UniStgGrid3D::bottom_bndy() const {
00270 return bottom_bndy_;
00271 }
00272
00273 mtk::Real mtk::UniStgGrid3D::top_bndy() const {
00274 return top_bndy_;
00275 }
00276
00277 int mtk::UniStgGrid3D::num_cells_z() const {
00278 return num_cells_z_;
00279 }
00280
00281 mtk::Real mtk::UniStgGrid3D::delta_z() const {
00282 return delta_z_;
00283 }
00284

```

```

00297
00298 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_z() const
00299 {
00300 return discrete_domain_z_.data();
00301 }
00302
00303 mtk::Real* mtk::UniStgGrid3D::discrete_field() {
00304 return discrete_field_.data();
00305 }
00306
00307
00308 bool mtk::UniStgGrid3D::Bound() const {
00309 return discrete_field_.size() != 0;
00310 }
00311
00312
00313 int mtk::UniStgGrid3D::Size() const {
00314 return discrete_field_.size();
00315 }
00316
00317
00318 void mtk::UniStgGrid3D::BindScalarField(
00319 mtk::Real (*ScalarField)(const mtk::Real &xx,
00320 const mtk::Real &yy,
00321 const mtk::Real &zz)) {
00322
00323 #ifdef MTK_PERFORM_PREVENTIONS
00324 mtk::Tools::Prevent(nature_ != mtk::SCALAR, __FILE__, __LINE__, __func__);
00325 #endif
00326
00327
00328
00329 discrete_domain_x_.reserve(num_cells_x_ + 2);
00330
00331 discrete_domain_x_.push_back(west_bndy_);
00332 #ifdef MTK_PRECISION_DOUBLE
00333 auto first_center = west_bndy_ + delta_x_/2.0;
00334 #else
00335 auto first_center = west_bndy_ + delta_x_/2.0f;
00336 #endif
00337 discrete_domain_x_.push_back(first_center);
00338 for (auto ii = 1; ii < num_cells_x_; ++ii) {
00339 discrete_domain_x_.push_back(first_center + ii*delta_x_);
00340 }
00341 discrete_domain_x_.push_back(east_bndy_);
00342
00343
00344
00345 discrete_domain_y_.reserve(num_cells_y_ + 2);
00346
00347 discrete_domain_y_.push_back(south_bndy_);
00348 #ifdef MTK_PRECISION_DOUBLE
00349 first_center = south_bndy_ + delta_x_/2.0;
00350 #else
00351 first_center = south_bndy_ + delta_x_/2.0f;
00352 #endif
00353 discrete_domain_y_.push_back(first_center);
00354 for (auto ii = 1; ii < num_cells_y_; ++ii) {
00355 discrete_domain_y_.push_back(first_center + ii*delta_y_);
00356 }
00357 discrete_domain_y_.push_back(north_bndy_);
00358
00359
00360
00361 discrete_domain_z_.reserve(num_cells_z_ + 2);
00362
00363 discrete_domain_z_.push_back(bottom_bndy_);
00364 first_center = bottom_bndy_ + delta_z_/mtk::kTwo;
00365 discrete_domain_z_.push_back(first_center);
00366 for (auto ii = 1; ii < num_cells_z_; ++ii) {
00367 discrete_domain_z_.push_back(first_center + ii*delta_z_);
00368 }
00369 discrete_domain_z_.push_back(top_bndy_);
00370
00371
00372
00373 int aux{(num_cells_x_ + 2)*(num_cells_y_ + 2)*(num_cells_z_ + 2)};
00374
00375 discrete_field_.reserve(aux);
00376
00377 for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00378 for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00379 for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00380 #if MTK_VERBOSE_LEVEL > 6

```



```

00381 std::cout << "At z = " << discrete_domain_z_[kk] << ": Pushing value"
00382 " for x = " << discrete_domain_x_[jj] << " y = " <<
00383 discrete_domain_y_[ii] << std::endl;
00384 #endif
00385 discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00386 discrete_domain_y_[ii],
00387 discrete_domain_z_[kk]));
00388 }
00389 }
00390 }
00391 }
00392
00393 void mtk::UniStgGrid3D::BindVectorFieldPComponent (
00394 mtk::Real (*VectorField) (const mtk::Real &xx,
00395 const mtk::Real &yy,
00396 const mtk::Real &zz)) {
00397 }
00398 }
00399
00400 void mtk::UniStgGrid3D::BindVectorFieldQComponent (
00401 mtk::Real (*VectorField) (const mtk::Real &xx,
00402 const mtk::Real &yy,
00403 const mtk::Real &zz)) {
00404 }
00405 }
00406
00407 void mtk::UniStgGrid3D::BindVectorFieldRComponent (
00408 mtk::Real (*VectorField) (const mtk::Real &xx,
00409 const mtk::Real &yy,
00410 const mtk::Real &zz)) {
00411 }
00412 }
00413
00414 void mtk::UniStgGrid3D::BindVectorField(
00415 mtk::Real (*VectorFieldPComponent) (const mtk::Real &xx,
00416 const mtk::Real &yy,
00417 const mtk::Real &zz),
00418 mtk::Real (*VectorFieldQComponent) (const mtk::Real &xx,
00419 const mtk::Real &yy,
00420 const mtk::Real &zz),
00421 mtk::Real (*VectorFieldRComponent) (const mtk::Real &xx,
00422 const mtk::Real &yy,
00423 const mtk::Real &zz)) {
00424
00425 #ifdef MTK_PERFORM_PREVENTIONS
00426 mtk::Tools::Prevent(nature_ != mtk::VECTOR, __FILE__, __LINE__, __func__);
00427 #endif
00428
00429 BindVectorFieldPComponent(VectorFieldPComponent);
00430 BindVectorFieldQComponent(VectorFieldQComponent);
00431 }
00432
00433 bool mtk::UniStgGrid3D::WriteToFile(std::string filename,
00434 std::string space_name_x,
00435 std::string space_name_y,
00436 std::string space_name_z,
00437 std::string field_name) const {
00438
00439 std::ofstream output_dat_file; // Output file.
00440
00441 output_dat_file.open(filename);
00442
00443 if (!output_dat_file.is_open()) {
00444 return false;
00445 }
00446
00447 if (nature_ == mtk::SCALAR) {
00448 output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00449 space_name_z << ' ' << field_name << std::endl;
00450
00451 int idx{};
00452 for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00453 for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00454 for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00455 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00456 discrete_domain_y_[ii] << ' ' << discrete_domain_z_[kk] << ' ' <<
00457 discrete_field_[idx] << std::endl;
00458 idx++;
00459 }
00460 }
00461 }

```

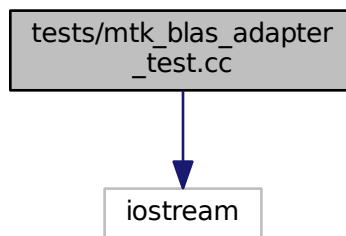
```
00462
00463 } else {
00464 output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00465 space_name_z << ' ' << field_name << std::endl;
00466 }
00467 }
00468
00469 output_dat_file.close();
00470
00471 return true;
00472 }
```

## 18.121 tests/mtk\_blas\_adapter\_test.cc File Reference

Test file for the [mtk::BLASAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for mtk\_blas\_adapter\_test.cc:



### Functions

- `int main ()`

#### 18.121.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_blas\\_adapter\\_test.cc](#).

#### 18.121.2 Function Documentation

##### 18.121.2.1 `int main ( )`

Definition at line [109](#) of file [mtk\\_blas\\_adapter\\_test.cc](#).

## 18.122 mtk\_blas\_adapter\_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestRealDenseMM() {
00061
00062 mtk::Tools::BeginUnitTestNo(1);
00063
00064 int rr = 2;
00065 int cc = 3;
00066
00067 mtk::DenseMatrix aa(rr,cc);
00068
00069 aa.SetValue(0,0,1.0);
00070 aa.SetValue(0,1,2.0);
00071 aa.SetValue(0,2,3.0);
00072 aa.SetValue(1,0,4.0);
00073 aa.SetValue(1,1,5.0);
00074 aa.SetValue(1,2,6.0);
00075
00076 mtk::DenseMatrix bb(cc,rr);
00077
00078 bb.SetValue(0,0,7.0);
00079 bb.SetValue(0,1,8.0);
00080 bb.SetValue(1,0,9.0);
00081 bb.SetValue(1,1,10.0);
00082 bb.SetValue(2,0,11.0);
00083 bb.SetValue(2,1,12.0);
00084 }
```

```

00085 mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00086
00087 mtk::DenseMatrix ff(rr,rr);
00088
00089 ff.SetValue(0,0,58.0);
00090 ff.SetValue(0,1,64.00);
00091 ff.SetValue(1,0,139.0);
00092 ff.SetValue(1,1,154.0);
00093
00094 mtk::Tools::EndUnitTestNo(1);
00095 mtk::Tools::Assert(pp == ff);
00096 }
00097
00098 int main () {
00099
00100 std::cout << "Testing mtk::BLASAdapter class." << std::endl;
00101
00102 TestRealDenseMM();
00103 }
00104
00105 #else
00106 #include <iostream>
00107 using std::cout;
00108 using std::endl;
00109 int main () {
00110 cout << "This code HAS to be compiled with support for C++11." << endl;
00111 cout << "Exiting..." << endl;
00112 }
00113 #endif

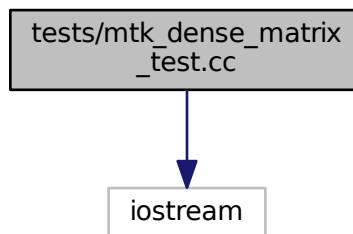
```

## 18.123 tests/mtk\_dense\_matrix\_test.cc File Reference

Test file for the `mtk::DenseMatrix` class.

```
#include <iostream>
```

Include dependency graph for `mtk_dense_matrix_test.cc`:



### Functions

- `int main ()`

#### 18.123.1 Detailed Description

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_dense\\_matrix\\_test.cc](#).

**18.123.2 Function Documentation****18.123.2.1 int main ( )**

Definition at line 349 of file [mtk\\_dense\\_matrix\\_test.cc](#).

**18.124 mtk\_dense\_matrix\_test.cc**

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062
00063 mtk::Tools::BeginUnitTestNo(1);
```

```

00064
00065 mtk::DenseMatrix m1;
00066
00067 mtk::Tools::EndUnitTestNo(1);
00068 mtk::Tools::Assert(m1.data() == nullptr);
00069 }
00070
00071 void TestConstructorWithNumRowsNumCols() {
00072
00073 mtk::Tools::BeginUnitTestNo(2);
00074
00075 int rr = 4;
00076 int cc = 7;
00077
00078 mtk::DenseMatrix m2(rr,cc);
00079
00080 mtk::Tools::EndUnitTestNo(2);
00081
00082 bool assertion =
00083 m2.data() != nullptr && m2.num_rows() == rr && m2.num_cols() == cc;
00084
00085 mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestConstructAsIdentity() {
00089
00090 mtk::Tools::BeginUnitTestNo(3);
00091
00092 int rank = 5;
00093 bool padded = true;
00094 bool transpose = false;
00095
00096 mtk::DenseMatrix m3(rank,padded,transpose);
00097
00098 mtk::DenseMatrix rr(rank + 2,rank);
00099
00100 for (int ii = 0; ii < rank; ++ii) {
00101 rr.SetValue(ii + 1, ii, mtk::kOne);
00102 }
00103
00104 mtk::Tools::EndUnitTestNo(3);
00105 mtk::Tools::Assert(m3 == rr);
00106 }
00107
00108 void TestConstructAsVandermonde() {
00109
00110 mtk::Tools::BeginUnitTestNo(4);
00111
00112 int rank = 5;
00113 bool padded = false;
00114 bool transpose = false;
00115
00116 mtk::DenseMatrix m4(rank,padded,transpose);
00117
00118 mtk::DenseMatrix rr(rank,rank);
00119
00120 for (int ii = 0; ii < rank; ++ii) {
00121 rr.SetValue(ii, ii, mtk::kOne);
00122 }
00123
00124 mtk::Tools::EndUnitTestNo(4);
00125 mtk::Tools::Assert(m4 == rr);
00126 }
00127
00128 void TestSetValueGetValue() {
00129
00130 mtk::Tools::BeginUnitTestNo(5);
00131
00132 int rr = 4;
00133 int cc = 7;
00134
00135 mtk::DenseMatrix m5(rr,cc);
00136
00137 for (auto ii = 0; ii < rr; ++ii) {
00138 for (auto jj = 0; jj < cc; ++jj) {
00139 m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00140 }
00141 }
00142
00143 mtk::Real *vals = m5.data();
00144

```

```

00145 bool assertion{true};
00146
00147 for (auto ii = 0; ii < rr && assertion; ++ii) {
00148 for (auto jj = 0; jj < cc && assertion; ++jj) {
00149 assertion = assertion && m5.GetValue(ii,jj) == vals[ii*cc + jj];
00150 }
00151 }
00152
00153 mtk::Tools::EndUnitTestNo(5);
00154 mtk::Tools::Assert(assertion);
00155 }
00156
00157 void TestConstructAsVandermondeTranspose() {
00158
00159 mtk::Tools::BeginUnitTestNo(6);
00160
00161 bool transpose = false;
00162 int generator_length = 3;
00163 int progression_length = 4;
00164
00165 mtk::Real generator[] = {-0.5, 0.5, 1.5};
00166
00167 mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00168
00169 transpose = true;
00170
00171 mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00172 mtk::DenseMatrix rr(progression_length, generator_length);
00173
00174 rr.SetValue(0, 0, 1.0);
00175 rr.SetValue(0, 1, 1.0);
00176 rr.SetValue(0, 2, 1.0);
00177
00178 rr.SetValue(1, 0, -0.5);
00179 rr.SetValue(1, 1, 0.5);
00180 rr.SetValue(1, 2, 1.5);
00181
00182 rr.SetValue(2, 0, 0.25);
00183 rr.SetValue(2, 1, 0.25);
00184 rr.SetValue(2, 2, 2.25);
00185
00186 rr.SetValue(3, 0, -0.125);
00187 rr.SetValue(3, 1, 0.125);
00188 rr.SetValue(3, 2, 3.375);
00189
00190 mtk::Tools::EndUnitTestNo(6);
00191 mtk::Tools::Assert(m7 == rr);
00192 }
00193
00194 void TestKron() {
00195
00196 mtk::Tools::BeginUnitTestNo(7);
00197
00198 bool padded = false;
00199 bool transpose = false;
00200 int lots_of_rows = 2;
00201 int lots_of_cols = 5;
00202 mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00203
00204 mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00205
00206 for (auto ii = 0; ii < lots_of_rows; ++ii) {
00207 for (auto jj = 0; jj < lots_of_cols; ++jj) {
00208 m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00209 }
00210 }
00211
00212 mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00213
00214 mtk::DenseMatrix rr(lots_of_rows*lots_of_rows, lots_of_rows*lots_of_cols);
00215
00216 rr.SetValue(0,0,1.0);
00217 rr.SetValue(0,1,2.0);
00218 rr.SetValue(0,2,3.0);
00219 rr.SetValue(0,3,4.0);
00220 rr.SetValue(0,4,5.0);
00221 rr.SetValue(0,5,0.0);
00222 rr.SetValue(0,6,0.0);
00223 rr.SetValue(0,7,0.0);
00224 rr.SetValue(0,8,0.0);
00225 rr.SetValue(0,9,0.0);

```

```

00226
00227 rr.SetValue(1,0,6.0);
00228 rr.SetValue(1,1,7.0);
00229 rr.SetValue(1,2,8.0);
00230 rr.SetValue(1,3,9.0);
00231 rr.SetValue(1,4,10.0);
00232 rr.SetValue(1,5,0.0);
00233 rr.SetValue(1,6,0.0);
00234 rr.SetValue(1,7,0.0);
00235 rr.SetValue(1,8,0.0);
00236 rr.SetValue(1,9,0.0);
00237
00238 rr.SetValue(2,0,0.0);
00239 rr.SetValue(2,1,0.0);
00240 rr.SetValue(2,2,0.0);
00241 rr.SetValue(2,3,0.0);
00242 rr.SetValue(2,4,0.0);
00243 rr.SetValue(2,5,1.0);
00244 rr.SetValue(2,6,2.0);
00245 rr.SetValue(2,7,3.0);
00246 rr.SetValue(2,8,4.0);
00247 rr.SetValue(2,9,5.0);
00248
00249 rr.SetValue(3,0,0.0);
00250 rr.SetValue(3,1,0.0);
00251 rr.SetValue(3,2,0.0);
00252 rr.SetValue(3,3,0.0);
00253 rr.SetValue(3,4,0.0);
00254 rr.SetValue(3,5,6.0);
00255 rr.SetValue(3,6,7.0);
00256 rr.SetValue(3,7,8.0);
00257 rr.SetValue(3,8,9.0);
00258 rr.SetValue(3,9,10.0);
00259
00260 mtk::Tools::EndUnitTestNo(7);
00261 mtk::Tools::Assert(m10 == rr);
00262 }
00263
00264 void TestConstructWithNumRowsNumColsAssignmentOperator() {
00265
00266 mtk::Tools::BeginUnitTestNo(8);
00267
00268 int lots_of_rows = 4;
00269 int lots_of_cols = 3;
00270 mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00271
00272 for (auto ii = 0; ii < lots_of_rows; ++ii) {
00273 for (auto jj = 0; jj < lots_of_cols; ++jj) {
00274 m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00275 }
00276 }
00277
00278 m11.Transpose();
00279
00280 mtk::DenseMatrix m12;
00281
00282 m12 = m11;
00283
00284 mtk::Tools::EndUnitTestNo(8);
00285 mtk::Tools::Assert(m11 == m12);
00286 }
00287
00288 void TestConstructAsVandermondeTransposeAssignmentOperator() {
00289
00290 mtk::Tools::BeginUnitTestNo(9);
00291
00292 bool transpose = false;
00293 int gg_l = 3;
00294 int progression_length = 4;
00295 mtk::Real gg[] = {-0.5, 0.5, 1.5};
00296
00297 mtk::DenseMatrix m13(gg, gg_l ,progression_length, transpose);
00298
00299 mtk::DenseMatrix m14;
00300
00301 m14 = m13;
00302
00303 m13.Transpose();
00304
00305 m14 = m13;
00306

```



```

00307 mtk::Tools::EndUnitTestNo(9);
00308 mtk::Tools::Assert(m13 == m14);
00309 }
00310
00311 void TestMaxFromSumsOfRowElements() {
00312
00313 mtk::Tools::BeginUnitTestNo(10);
00314
00315 mtk::DenseMatrix mm(3, 4);
00316
00317 for (int ii = 0; ii < mm.num_rows(); ++ii) {
00318 for (int jj = 0; jj < mm.num_cols(); ++jj) {
00319 mm.SetValue(ii, jj, mtk::kOne);
00320 }
00321 }
00322
00323 bool assertion{mm.MaxFromSumsOfRowElements() == 4};
00324
00325 mtk::Tools::EndUnitTestNo(10);
00326 mtk::Tools::Assert(assertion);
00327 }
00328
00329 int main () {
00330
00331 std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00332
00333 TestDefaultConstructor();
00334 TestConstructorWithNumRowsNumCols();
00335 TestConstructAsIdentity();
00336 TestConstructAsVandermonde();
00337 TestSetValueGetValue();
00338 TestConstructAsVandermondeTranspose();
00339 TestKron();
00340 TestConstructWithNumRowsNumColsAssignmentOperator();
00341 TestConstructAsVandermondeTransposeAssignmentOperator();
00342 TestMaxFromSumsOfRowElements();
00343 }
00344
00345 #else
00346 #include <iostream>
00347 using std::cout;
00348 using std::endl;
00349 int main () {
00350 cout << "This code HAS to be compiled with support for C++11." << endl;
00351 cout << "Exiting..." << endl;
00352 }
00353 #endif

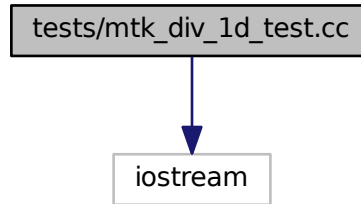
```

## 18.125 tests/mtk\_div\_1d\_test.cc File Reference

Testing the mimetic 1D divergence, constructed with the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for `mtk_div_1d_test.cc`:



## Functions

- `int main ()`

### 18.125.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)

Definition in file [mtk\\_div\\_1d\\_test.cc](#).

### 18.125.2 Function Documentation

#### 18.125.2.1 `int main ( )`

Definition at line [288](#) of file [mtk\\_div\\_1d\\_test.cc](#).

## 18.126 `mtk_div_1d_test.cc`

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```

```
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <iostream>
00056 #include "mtk.h"
00057
00058 void TestDefaultConstructorFactoryMethodDefault() {
00059
00060 mtk::Tools::BeginUnitTestNo(1);
00061
00062 mtk::Div1D div2;
00063
00064 bool assertion = div2.ConstructDiv1D();
00065
00066 if (!assertion) {
00067 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00068 }
00069
00070 mtk::Tools::EndUnitTestNo(1);
00071 mtk::Tools::Assert(assertion);
00072 }
00073
00074 void TestDefaultConstructorFactoryMethodFourthOrder() {
00075
00076 mtk::Tools::BeginUnitTestNo(2);
00077
00078 mtk::Div1D div4;
00079
00080 bool assertion = div4.ConstructDiv1D(4);
00081
00082 if (!assertion) {
00083 std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00084 }
00085
00086 mtk::Tools::EndUnitTestNo(2);
00087 mtk::Tools::Assert(assertion);
00088 }
00089
00090 void TestDefaultConstructorFactoryMethodSixthOrder() {
00091
00092 mtk::Tools::BeginUnitTestNo(3);
00093
00094 mtk::Div1D div6;
00095
00096 bool assertion = div6.ConstructDiv1D(6);
00097
00098 if (!assertion) {
00099 std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00100 }
00101
00102 mtk::Tools::EndUnitTestNo(3);
00103 mtk::Tools::Assert(assertion);
00104 }
00105
```

```

00106 }
00107
00108 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00109 mtk::Tools::BeginUnitTestNo(4);
00110
00111 mtk::Div1D div8;
00112
00113 bool assertion = div8.ConstructDiv1D(8);
00114
00115 if (!assertion) {
00116 std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00117 }
00118
00119 mtk::Tools::EndUnitTestNo(4);
00120 mtk::Tools::Assert(assertion);
00121 }
00122
00123 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00124 mtk::Tools::BeginUnitTestNo(5);
00125
00126 mtk::Div1D div10;
00127
00128 bool assertion = div10.ConstructDiv1D(10);
00129
00130 if (!assertion) {
00131 std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00132 }
00133
00134 mtk::Tools::EndUnitTestNo(5);
00135 mtk::Tools::Assert(assertion);
00136 }
00137
00138 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00139 mtk::Tools::BeginUnitTestNo(6);
00140
00141 mtk::Div1D div12;
00142
00143 bool assertion = div12.ConstructDiv1D(12);
00144
00145 if (!assertion) {
00146 std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00147 }
00148
00149 mtk::Tools::EndUnitTestNo(6);
00150 mtk::Tools::Assert(assertion);
00151 }
00152
00153 void TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold() {
00154 mtk::Tools::BeginUnitTestNo(7);
00155
00156 mtk::Div1D div14;
00157
00158 bool assertion = div14.ConstructDiv1D(14);
00159
00160 if (!assertion) {
00161 std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00162 }
00163
00164 mtk::Tools::EndUnitTestNo(7);
00165 mtk::Tools::Assert(assertion);
00166 }
00167
00168 void TestSecondOrderReturnAsDenseMatrixWithGrid() {
00169 mtk::Tools::BeginUnitTestNo(8);
00170
00171 mtk::Div1D div2;
00172
00173 bool assertion = div2.ConstructDiv1D();
00174
00175 if (!assertion) {
00176 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00177 }
00178
00179 mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00180
00181 mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00182 }
00183
00184
00185
00186

```

```

00187
00188 int rr{7};
00189 int cc{6};
00190
00191 mtk::DenseMatrix ref(rr, cc);
00192
00193 // Row 2.
00194 ref.SetValue(1,0,-5.0);
00195 ref.SetValue(1,1,5.0);
00196 ref.SetValue(1,2,0.0);
00197 ref.SetValue(1,3,0.0);
00198 ref.SetValue(1,4,0.0);
00199 ref.SetValue(1,5,0.0);
00200 ref.SetValue(1,6,0.0);
00201
00202 // Row 3.
00203 ref.SetValue(2,0,0.0);
00204 ref.SetValue(2,1,-5.0);
00205 ref.SetValue(2,2,5.0);
00206 ref.SetValue(2,3,0.0);
00207 ref.SetValue(2,4,0.0);
00208 ref.SetValue(2,5,0.0);
00209 ref.SetValue(2,6,0.0);
00210
00211 // Row 4.
00212 ref.SetValue(3,0,0.0);
00213 ref.SetValue(3,1,0.0);
00214 ref.SetValue(3,2,-5.0);
00215 ref.SetValue(3,3,5.0);
00216 ref.SetValue(3,4,0.0);
00217 ref.SetValue(3,5,0.0);
00218 ref.SetValue(3,6,0.0);
00219
00220 // Row 5.
00221 ref.SetValue(4,0,0.0);
00222 ref.SetValue(4,1,0.0);
00223 ref.SetValue(4,2,0.0);
00224 ref.SetValue(4,3,-5.0);
00225 ref.SetValue(4,4,5.0);
00226 ref.SetValue(4,5,0.0);
00227 ref.SetValue(4,6,0.0);
00228
00229 // Row 6.
00230 ref.SetValue(5,0,0.0);
00231 ref.SetValue(5,1,0.0);
00232 ref.SetValue(5,2,0.0);
00233 ref.SetValue(5,3,0.0);
00234 ref.SetValue(5,4,-5.0);
00235 ref.SetValue(5,5,5.0);
00236 ref.SetValue(5,6,0.0);
00237
00238 assertion = assertion && (div2m == ref);
00239
00240 mtk::Tools::EndUnitTestNo(8);
00241 mtk::Tools::Assert(assertion);
00242 }
00243
00244 void TestFourthOrderReturnAsDenseMatrixWithGrid() {
00245
00246 mtk::Tools::BeginUnitTestNo(9);
00247
00248 mtk::Div1D div4;
00249
00250 bool assertion = div4.ConstructDiv1D(4);
00251
00252 if (!assertion) {
00253 std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00254 }
00255
00256 std::cout << div4 << std::endl;
00257
00258 mtk::UniStgGrid1D grid(0.0, 1.0, 11);
00259
00260 std::cout << grid << std::endl;
00261
00262 mtk::DenseMatrix div4m(div4.ReturnAsDenseMatrix(grid));
00263
00264 std::cout << div4m << std::endl;
00265
00266 mtk::Tools::EndUnitTestNo(9);
00267 }

```

```

00268
00269 int main () {
00270
00271 std::cout << "Testing mtk::Div1D class." << std::endl;
00272
00273 TestDefaultConstructorFactoryMethodDefault();
00274 TestDefaultConstructorFactoryMethodFourthOrder();
00275 TestDefaultConstructorFactoryMethodSixthOrder();
00276 TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00277 TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00278 TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold();
00279 TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold();
00280 TestSecondOrderReturnAsDenseMatrixWithGrid();
00281 TestFourthOrderReturnAsDenseMatrixWithGrid();
00282 }
00283
00284 #else
00285 #include <iostream>
00286 using std::cout;
00287 using std::endl;
00288 int main () {
00289 cout << "This code HAS to be compiled with support for C++11." << endl;
00290 cout << "Exiting..." << endl;
00291 }
00292 #endif

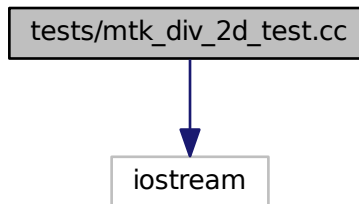
```

## 18.127 tests/mtk\_div\_2d\_test.cc File Reference

Test file for the [mtk::Div2D](#) class.

```
#include <iostream>
```

Include dependency graph for mtk\_div\_2d\_test.cc:



### Functions

- [int main \(\)](#)

#### 18.127.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_2d\\_test.cc](#).

## 18.127.2 Function Documentation

### 18.127.2.1 int main ( )

Definition at line 139 of file [mtk\\_div\\_2d\\_test.cc](#).

## 18.128 mtk\_div\_2d\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::Div2D dd;
00068
00069 mtk::Real aa = 0.0;
00070 mtk::Real bb = 1.0;
00071 mtk::Real cc = 0.0;
00072 mtk::Real ee = 1.0;
00073

```

```

00074 int nn = 5;
00075 int mm = 5;
00076
00077 mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00078
00079 bool assertion = dd.ConstructDiv2D(ddg);
00080
00081 if (!assertion) {
00082 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00083 }
00084
00085 mtk::Tools::EndUnitTestNo(1);
00086 mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091 mtk::Tools::BeginUnitTestNo(2);
00092
00093 mtk::Div2D dd;
00094
00095 mtk::Real aa = 0.0;
00096 mtk::Real bb = 1.0;
00097 mtk::Real cc = 0.0;
00098 mtk::Real ee = 1.0;
00099
00100 int nn = 5;
00101 int mm = 5;
00102
00103 mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00104
00105 bool assertion = dd.ConstructDiv2D(ddg);
00106
00107 if (!assertion) {
00108 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00109 }
00110
00111 mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00112
00113 assertion = assertion && (ddm.num_rows() != mtk::kZero);
00114
00115 std::cout << ddm << std::endl;
00116
00117 assertion = assertion && ddm.WriteToFile("mtk_div_2d_test_02.dat");
00118
00119 if (!assertion) {
00120 std::cerr << "Error writing to file." << std::endl;
00121 }
00122
00123 mtk::Tools::EndUnitTestNo(2);
00124 mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129 std::cout << "Testing mtk::Div2D class." << std::endl;
00130
00131 TestDefaultConstructorFactory();
00132 TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140 cout << "This code HAS to be compiled with support for C++11." << endl;
00141 cout << "Exiting..." << endl;
00142 }
00143 #endif

```

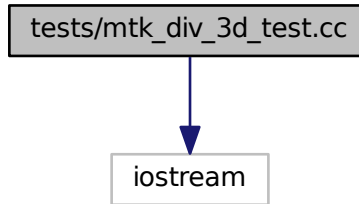
## 18.129 tests/mtk\_div\_3d\_test.cc File Reference

Test file for the `mtk::Div3D` class.



```
#include <iostream>
```

Include dependency graph for mtk\_div\_3d\_test.cc:



## Functions

- int `main` ()

### 18.129.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_div\\_3d\\_test.cc](#).

### 18.129.2 Function Documentation

#### 18.129.2.1 int main ( )

Definition at line 145 of file [mtk\\_div\\_3d\\_test.cc](#).

## 18.130 mtk\_div\_3d\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,

```

```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <cmath>
00056 #include <ctime>
00057 #include <iostream>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactory() {
00061 mtk::Tools::BeginUnitTestNo(1);
00062 mtk::Div3D div;
00063 mtk::Real aa = 0.0;
00064 mtk::Real bb = 1.0;
00065 mtk::Real cc = 0.0;
00066 mtk::Real dd = 1.0;
00067 mtk::Real ee = 0.0;
00068 mtk::Real ff = 1.0;
00069 int nn = 5;
00070 int mm = 5;
00071 int oo = 5;
00072 mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00073 bool assertion = div.ConstructDiv3D(divg);
00074 if (!assertion) {
00075 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00076 }
00077 mtk::Tools::EndUnitTestNo(1);
00078 mtk::Tools::Assert(assertion);
00079 }
00080
00081 void TestReturnAsDenseMatrixWriteToFile() {
00082 mtk::Tools::BeginUnitTestNo(2);
00083 mtk::Div3D div;
00084 mtk::Real aa = 0.0;
00085 mtk::Real bb = 1.0;
00086 mtk::Real cc = 0.0;
00087 mtk::Real dd = 1.0;
00088 mtk::Real ee = 0.0;
00089 mtk::Real ff = 1.0;
00090 int nn = 5;

```

```

00106 int mm = 5;
00107 int oo = 5;
00108
00109 mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00110
00111 bool assertion = div.ConstructDiv3D(divg);
00112
00113 if (!assertion) {
00114 std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00115 }
00116
00117 mtk::DenseMatrix divm(div.ReturnAsDenseMatrix());
00118
00119 assertion = assertion && (divm.num_rows() != mtk::kZero);
00120
00121 std::cout << divm << std::endl;
00122
00123 assertion = assertion && divm.WriteToFile("mtk_div_3d_test_02.dat");
00124
00125 if (!assertion) {
00126 std::cerr << "Error writing to file." << std::endl;
00127 }
00128
00129 mtk::Tools::EndUnitTestNo(2);
00130 mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135 std::cout << "Testing mtk::Div3D class." << std::endl;
00136
00137 TestDefaultConstructorFactory();
00138 TestReturnAsDenseMatrixWriteToFile();
00139 }
00140
00141 #else
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146 cout << "This code HAS to be compiled with support for C++11." << endl;
00147 cout << "Exiting..." << endl;
00148 }
00149 #endif

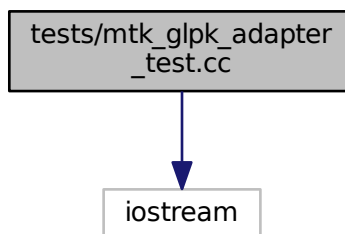
```

## 18.131 tests/mtk\_glpk\_adapter\_test.cc File Reference

Test file for the [mtk::GLPKAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_glpk_adapter_test.cc`:



## Functions

- `int main ()`

### 18.131.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Test the `mtk::GLPKAdapter` class.

Definition in file `mtk_glpk_adapter_test.cc`.

### 18.131.2 Function Documentation

#### 18.131.2.1 `int main ( )`

Definition at line 81 of file `mtk_glpk_adapter_test.cc`.

## 18.132 `mtk_glpk_adapter_test.cc`

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

```

```

00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072 std::cout << "Testing mtk::GLPKAdapter class." << std::endl;
00073
00074 Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082 cout << "This code HAS to be compiled with support for C++11." << endl;
00083 cout << "Exiting..." << endl;
00084 }
00085 #endif

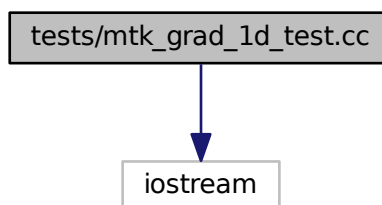
```

## 18.133 tests/mtk\_grad\_1d\_test.cc File Reference

Testing the mimetic 1D gradient, constructed with the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for mtk\_grad\_1d\_test.cc:



## Functions

- int `main` ()

### 18.133.1 Detailed Description

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_1d\\_test.cc](#).

**18.133.2 Function Documentation****18.133.2.1 int main ( )**

Definition at line 319 of file [mtk\\_grad\\_1d\\_test.cc](#).

**18.134 mtk\_grad\_1d\_test.cc**

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062 mtk::Tools::BeginUnitTestNo(1);
00063

```

```

00064 mtk::Grad1D grad2;
00065
00066 bool assertion = grad2.ConstructGrad1D();
00067
00068 if (!assertion) {
00069 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00070 }
00071
00072
00073 std::cout << grad2 << std::endl;
00074
00075 mtk::Tools::EndUnitTestNo(1);
00076 mtk::Tools::Assert(assertion);
00077 }
00078
00079 void TestDefaultConstructorFactoryMethodFourthOrder() {
00080
00081 mtk::Tools::BeginUnitTestNo(2);
00082
00083 mtk::Grad1D grad4;
00084
00085 bool assertion = grad4.ConstructGrad1D(4);
00086
00087 if (!assertion) {
00088 std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00089 }
00090
00091 std::cout << grad4 << std::endl;
00092
00093 mtk::Tools::EndUnitTestNo(2);
00094 mtk::Tools::Assert(assertion);
00095 }
00096
00097 void TestDefaultConstructorFactoryMethodSixthOrder() {
00098
00099 mtk::Tools::BeginUnitTestNo(3);
00100
00101 mtk::Grad1D grad6;
00102
00103 bool assertion = grad6.ConstructGrad1D(6);
00104
00105 if (!assertion) {
00106 std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00107 }
00108
00109 std::cout << grad6 << std::endl;
00110
00111 mtk::Tools::EndUnitTestNo(3);
00112 mtk::Tools::Assert(assertion);
00113 }
00114
00115 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00116
00117 mtk::Tools::BeginUnitTestNo(4);
00118
00119 mtk::Grad1D grad8;
00120
00121 bool assertion = grad8.ConstructGrad1D(8);
00122
00123 if (!assertion) {
00124 std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00125 }
00126
00127 std::cout << grad8 << std::endl;
00128
00129 mtk::Tools::EndUnitTestNo(4);
00130 mtk::Tools::Assert(assertion);
00131 }
00132
00133 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00134
00135 mtk::Tools::BeginUnitTestNo(5);
00136
00137 mtk::Grad1D grad10;
00138
00139 bool assertion = grad10.ConstructGrad1D(10);
00140
00141 if (!assertion) {
00142 std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00143 }
00144

```

```

00145 std::cout << grad10 << std::endl;
00146
00147 mtk::Tools::EndUnitTestNo(5);
00148 mtk::Tools::Assert(assertion);
00149 }
00150
00151 void TestReturnAsDenseMatrixWithGrid() {
00152
00153 mtk::Tools::BeginUnitTestNo(6);
00154
00155 mtk::Grad1D grad2;
00156
00157 bool assertion = grad2.ConstructGrad1D();
00158
00159 if (!assertion) {
00160 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00161 }
00162
00163 mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00164
00165 mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00166
00167 int rr{6};
00168 int cc{7};
00169
00170 mtk::DenseMatrix ref(rr, cc);
00171
00172 // Row 1.
00173 ref.SetValue(0,0,-13.3333);
00174 ref.SetValue(0,1,15);
00175 ref.SetValue(0,2,-1.66667);
00176 ref.SetValue(0,3,0.0);
00177 ref.SetValue(0,4,0.0);
00178 ref.SetValue(0,5,0.0);
00179 ref.SetValue(0,6,0.0);
00180
00181 // Row 2.
00182 ref.SetValue(1,0,0.0);
00183 ref.SetValue(1,1,-5.0);
00184 ref.SetValue(1,2,5.0);
00185 ref.SetValue(1,3,0.0);
00186 ref.SetValue(1,4,0.0);
00187 ref.SetValue(1,5,0.0);
00188 ref.SetValue(1,6,0.0);
00189
00190 // Row 3.
00191 ref.SetValue(2,0,0.0);
00192 ref.SetValue(2,1,0.0);
00193 ref.SetValue(2,2,-5.0);
00194 ref.SetValue(2,3,5.0);
00195 ref.SetValue(2,4,0.0);
00196 ref.SetValue(2,5,0.0);
00197 ref.SetValue(2,6,0.0);
00198
00199 // Row 4.
00200 ref.SetValue(3,0,0.0);
00201 ref.SetValue(3,1,0.0);
00202 ref.SetValue(3,2,0.0);
00203 ref.SetValue(3,3,-5.0);
00204 ref.SetValue(3,4,5.0);
00205 ref.SetValue(3,5,0.0);
00206 ref.SetValue(3,6,0.0);
00207
00208 // Row 5.
00209 ref.SetValue(4,0,0.0);
00210 ref.SetValue(4,1,0.0);
00211 ref.SetValue(4,2,0.0);
00212 ref.SetValue(4,3,0.0);
00213 ref.SetValue(4,4,-5.0);
00214 ref.SetValue(4,5,5.0);
00215 ref.SetValue(4,6,0.0);
00216
00217 // Row 6.
00218 ref.SetValue(5,0,0.0);
00219 ref.SetValue(5,1,0.0);
00220 ref.SetValue(5,2,0.0);
00221 ref.SetValue(5,3,0.0);
00222 ref.SetValue(5,4,1.66667);
00223 ref.SetValue(5,5,-15.0);
00224 ref.SetValue(5,6,13.3333);
00225

```



```

00226 mtk::Tools::EndUnitTestNo(6);
00227 mtk::Tools::Assert(grad2m == ref);
00228 }
00229
00230 void TestReturnAsDimensionlessDenseMatrix() {
00231
00232 mtk::Tools::BeginUnitTestNo(7);
00233
00234 mtk::Grad1D grad4;
00235
00236 bool assertion = grad4.ConstructGrad1D(4);
00237
00238 if (!assertion) {
00239 std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00240 }
00241
00242 mtk::DenseMatrix grad4m(grad4.ReturnAsDimensionlessDenseMatrix
00243 (10));
00244
00245 std::cout << grad4m << std::endl;
00246
00247 mtk::Tools::EndUnitTestNo(7);
00248 mtk::Tools::Assert(assertion);
00249 }
00250
00251 void TestWriteToFile() {
00252
00253 mtk::Tools::BeginUnitTestNo(8);
00254
00255 mtk::Grad1D grad2;
00256
00257 bool assertion = grad2.ConstructGrad1D();
00258
00259 if (!assertion) {
00260 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00261 }
00262
00263 mtk::UniStgGrid1D grid(0.0, 1.0, 50);
00264
00265 mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00266
00267 std::cout << grad2m << std::endl;
00268
00269 assertion = assertion && grad2m.WriteToFile("mtk_grad_1d_test_08.dat");
00270
00271 if (!assertion) {
00272 std::cerr << "Error writing to file." << std::endl;
00273 }
00274
00275 mtk::Tools::EndUnitTestNo(8);
00276 mtk::Tools::Assert(assertion);
00277 }
00278
00279 void TestMimBndy() {
00280
00281 mtk::Tools::BeginUnitTestNo(9);
00282
00283 mtk::Grad1D grad2;
00284
00285 bool assertion = grad2.ConstructGrad1D();
00286
00287 if (!assertion) {
00288 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00289 }
00290
00291 std::cout << grad2 << std::endl;
00292
00293 mtk::DenseMatrix grad2m(grad2.mim_bndy());
00294
00295 std::cout << grad2m << std::endl;
00296
00297 mtk::Tools::EndUnitTestNo(9);
00298 mtk::Tools::Assert(assertion);
00299 }
00300
00301 int main () {
00302
00303 std::cout << "Testing mtk::Grad1D class." << std::endl;
00304
00305 TestDefaultConstructorFactoryMethodDefault();
00306 TestDefaultConstructorFactoryMethodFourthOrder();

```

```

00306 TestDefaultConstructorFactoryMethodSixthOrder();
00307 TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00308 TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00309 TestReturnAsDenseMatrixWithGrid();
00310 TestReturnAsDimensionlessDenseMatrix();
00311 TestWriteToFile();
00312 TestMimBndy();
00313 }
00314
00315 #else
00316 #include <iostream>
00317 using std::cout;
00318 using std::endl;
00319 int main () {
00320 cout << "This code HAS to be compiled with support for C++11." << endl;
00321 cout << "Exiting..." << endl;
00322 }
00323 #endif

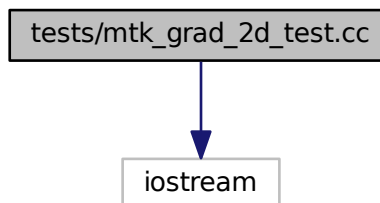
```

## 18.135 tests/mtk\_grad\_2d\_test.cc File Reference

Test file for the [mtk::Grad2D](#) class.

```
#include <iostream>
```

Include dependency graph for mtk\_grad\_2d\_test.cc:



### Functions

- int [main](#) ()

### 18.135.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_2d\\_test.cc](#).

### 18.135.2 Function Documentation

## 18.135.2.1 int main ( )

Definition at line 139 of file [mtk\\_grad\\_2d\\_test.cc](#).

## 18.136 mtk\_grad\_2d\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::Grad2D gg;
00068
00069 mtk::Real aa = 0.0;
00070 mtk::Real bb = 1.0;
00071 mtk::Real cc = 0.0;
00072 mtk::Real dd = 1.0;
00073
00074 int nn = 5;
00075 int mm = 5;
00076

```

```

00077 mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00078
00079 bool assertion = gg.ConstructGrad2D(ggg);
00080
00081 if (!assertion) {
00082 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00083 }
00084
00085 mtk::Tools::EndUnitTestNo(1);
00086 mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091 mtk::Tools::BeginUnitTestNo(2);
00092
00093 mtk::Grad2D gg;
00094
00095 mtk::Real aa = 0.0;
00096 mtk::Real bb = 1.0;
00097 mtk::Real cc = 0.0;
00098 mtk::Real dd = 1.0;
00099
00100 int nn = 5;
00101 int mm = 5;
00102
00103 mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00104
00105 bool assertion = gg.ConstructGrad2D(ggg);
00106
00107 if (!assertion) {
00108 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00109 }
00110
00111 mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00112
00113 assertion = assertion && (ggm.num_rows() != mtk::kZero);
00114
00115 std::cout << ggm << std::endl;
00116
00117 assertion = assertion && ggm.WriteToFile("mtk_grad_2d_test_02.dat");
00118
00119 if(!assertion) {
00120 std::cerr << "Error writing to file." << std::endl;
00121 }
00122
00123 mtk::Tools::EndUnitTestNo(2);
00124 mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129 std::cout << "Testing mtk::Grad2D class." << std::endl;
00130
00131 TestDefaultConstructorFactory();
00132 TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140 cout << "This code HAS to be compiled with support for C++11." << endl;
00141 cout << "Exiting..." << endl;
00142 }
00143 #endif

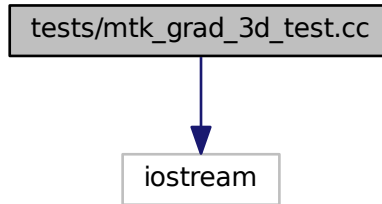
```

## 18.137 tests/mtk\_grad\_3d\_test.cc File Reference

Test file for the `mtk::Grad3D` class.

```
#include <iostream>
```

Include dependency graph for mtk\_grad\_3d\_test.cc:



## Functions

- `int main ()`

### 18.137.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_grad\\_3d\\_test.cc](#).

### 18.137.2 Function Documentation

#### 18.137.2.1 `int main ( )`

Definition at line 145 of file [mtk\\_grad\\_3d\\_test.cc](#).

## 18.138 mtk\_grad\_3d\_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```

```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <cmath>
00056 #include <ctime>
00057 #include <iostream>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactory() {
00061 mtk::Tools::BeginUnitTestNo(1);
00062
00063 mtk::Grad3D gg;
00064
00065 mtk::Real aa = 0.0;
00066 mtk::Real bb = 1.0;
00067 mtk::Real cc = 0.0;
00068 mtk::Real dd = 1.0;
00069 mtk::Real ee = 0.0;
00070 mtk::Real ff = 1.0;
00071
00072 int nn = 5;
00073 int mm = 5;
00074 int oo = 5;
00075
00076 mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo, mtk::VECTOR);
00077
00078 bool assertion = gg.ConstructGrad3D(ggg);
00079
00080 if (!assertion) {
00081 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00082 }
00083
00084 mtk::Tools::EndUnitTestNo(1);
00085 mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestReturnAsDenseMatrixWriteToFile() {
00089 mtk::Tools::BeginUnitTestNo(2);
00090
00091 mtk::Grad3D gg;
00092
00093 mtk::Real aa = 0.0;
00094 mtk::Real bb = 1.0;
00095 mtk::Real cc = 0.0;
00096 mtk::Real dd = 1.0;
00097 mtk::Real ee = 0.0;
00098 mtk::Real ff = 1.0;
00099
00100 int nn = 5;

```

```

00106 int mm = 5;
00107 int oo = 5;
00108
00109 mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo, mtk::VECTOR);
00110
00111 bool assertion = gg.ConstructGrad3D(ggg);
00112
00113 if (!assertion) {
00114 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00115 }
00116
00117 mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00118
00119 assertion = assertion && (ggm.num_rows() != mtk::kZero);
00120
00121 std::cout << ggm << std::endl;
00122
00123 assertion = assertion && ggm.WriteToFile("mtk_grad_3d_test_02.dat");
00124
00125 if (!assertion) {
00126 std::cerr << "Error writing to file." << std::endl;
00127 }
00128
00129 mtk::Tools::EndUnitTestNo(2);
00130 mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135 std::cout << "Testing mtk::Grad2D class." << std::endl;
00136
00137 TestDefaultConstructorFactory();
00138 TestReturnAsDenseMatrixWriteToFile();
00139 }
00140
00141 #else
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146 cout << "This code HAS to be compiled with support for C++11." << endl;
00147 cout << "Exiting..." << endl;
00148 }
00149 #endif

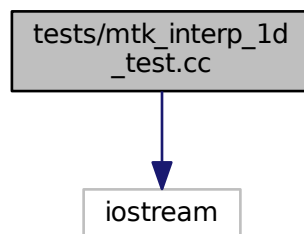
```

## 18.139 tests/mtk\_interp\_1d\_test.cc File Reference

Testing the 1D interpolation.

```
#include <iostream>
```

Include dependency graph for mtk\_interp\_1d\_test.cc:



## Functions

- `int main ()`

### 18.139.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu  
: Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file [mtk\\_interp\\_1d\\_test.cc](#).

### 18.139.2 Function Documentation

#### 18.139.2.1 `int main ( )`

Definition at line 113 of file [mtk\\_interp\\_1d\\_test.cc](#).

## 18.140 `mtk_interp_1d_test.cc`

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```



```

00054 */
00055
00056 #if __cplusplus == 201103L
00057 #include <iostream>
00059 #include "mtk.h"
00061
00062 void TestDefaultConstructorFactoryMethodDefault() {
00063 mtk::Tools::BeginUnitTestNo(1);
00065 mtk::InterplD inter;
00067 bool assertion = inter.ConstructInterplD();
00069 if (!assertion) {
00070 std::cerr << "Mimetic interp could not be built." << std::endl;
00072 }
00073
00074 mtk::Tools::EndUnitTestNo(1);
00075 mtk::Tools::Assert(assertion);
00076 }
00077
00078 void TestReturnAsDenseMatrixWithGrid() {
00079 mtk::Tools::BeginUnitTestNo(2);
00081 mtk::InterplD inter;
00083 bool assertion = inter.ConstructInterplD();
00085 if (!assertion) {
00086 std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00088 }
00089
00090 mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00091
00092 mtk::DenseMatrix interpm(inter.ReturnAsDenseMatrix(grid));
00093
00094 assertion =
00095 assertion && interpm.GetValue(0,0) == 1.0 && interpm.GetValue(5,6) == 1.0;
00096
00097 mtk::Tools::EndUnitTestNo(2);
00098 mtk::Tools::Assert(assertion);
00099 }
00100
00101 int main () {
00102 std::cout << "Testing mtk::InterplD class." << std::endl;
00104 TestDefaultConstructorFactoryMethodDefault();
00106 TestReturnAsDenseMatrixWithGrid();
00107 }
00108
00109 #else
00110 #include <iostream>
00111 using std::cout;
00112 using std::endl;
00113 int main () {
00114 cout << "This code HAS to be compiled with support for C++11." << endl;
00115 cout << "Exiting..." << endl;
00116 }
00117 #endif

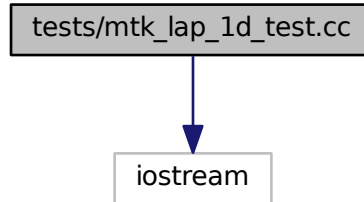
```

## 18.141 tests/mtk\_lap\_1d\_test.cc File Reference

Testing the 1D Laplacian operator.

```
#include <iostream>
```

Include dependency graph for `mtk_lap_1d_test.cc`:



## Functions

- `int main ()`

### 18.141.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)  
: Johnny Corbino - [jcorbino at mail dot sdsu dot edu](mailto:jcorbino@mail.sdsu.edu)

Definition in file [mtk\\_lap\\_1d\\_test.cc](#).

### 18.141.2 Function Documentation

#### 18.141.2.1 `int main ( )`

Definition at line [193](#) of file [mtk\\_lap\\_1d\\_test.cc](#).

## 18.142 `mtk_lap_1d_test.cc`

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
```

```
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059
00060 #include "mtk.h"
00061
00062 void TestDefaultConstructorFactoryMethodDefault() {
00063
00064 mtk::Tools::BeginUnitTestNo(1);
00065
00066 mtk::Lap1D lap2;
00067
00068 bool assertion = lap2.ConstructLap1D();
00069
00070 if (!assertion) {
00071 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00072 }
00073
00074 mtk::Tools::EndUnitTestNo(1);
00075 mtk::Tools::Assert(assertion);
00076 }
00077
00078 void TestDefaultConstructorFactoryMethodFourthOrder() {
00079
00080 mtk::Tools::BeginUnitTestNo(2);
00081
00082 mtk::Lap1D lap4;
00083
00084 bool assertion = lap4.ConstructLap1D(4);
00085
00086 if (!assertion) {
00087 std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00088 }
00089
00090 mtk::Tools::EndUnitTestNo(2);
00091 mtk::Tools::Assert(assertion);
00092 }
00093
00094 void TestDefaultConstructorFactoryMethodSixthOrder() {
00095
00096 mtk::Tools::BeginUnitTestNo(3);
00097
00098 mtk::Lap1D lap6;
00099
00100 bool assertion = lap6.ConstructLap1D(6);
00101
00102 if (!assertion) {
00103 std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00104 }
00105
00106 mtk::Tools::EndUnitTestNo(3);
```

```

00107 mtk::Tools::Assert(assertion);
00108 }
00109
00110 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00111
00112 mtk::Tools::BeginUnitTestNo(4);
00113
00114 mtk::Lap1D lap8;
00115
00116 bool assertion = lap8.ConstructLap1D(8);
00117
00118 if (!assertion) {
00119 std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00120 }
00121
00122 mtk::Tools::EndUnitTestNo(4);
00123 }
00124
00125 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00126
00127 mtk::Tools::BeginUnitTestNo(5);
00128
00129 mtk::Lap1D lap10;
00130
00131 bool assertion = lap10.ConstructLap1D(10);
00132
00133 if (!assertion) {
00134 std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00135 }
00136
00137 mtk::Tools::EndUnitTestNo(5);
00138 }
00139
00140 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00141
00142 mtk::Tools::BeginUnitTestNo(6);
00143
00144 mtk::Lap1D lap12;
00145
00146 bool assertion = lap12.ConstructLap1D(12);
00147
00148 if (!assertion) {
00149 std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00150 }
00151
00152 mtk::Tools::EndUnitTestNo(6);
00153 }
00154
00155 void TestReturnAsDenseMatrix() {
00156
00157 mtk::Tools::BeginUnitTestNo(8);
00158
00159 mtk::Lap1D lap4;
00160
00161 bool assertion = lap4.ConstructLap1D(4);
00162
00163 if (!assertion) {
00164 std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00165 }
00166
00167 mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00168
00169 mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00170
00171 assertion = assertion &&
00172 abs(lap4_m.GetValue(1, 0) - 385.133) < mtk::kDefaultTolerance &&
00173 abs(lap4_m.GetValue(11, 12) - 385.133) < mtk::kDefaultTolerance;
00174 mtk::Tools::EndUnitTestNo(8);
00175 mtk::Tools::Assert(assertion);
00176 }
00177
00178 int main () {
00179
00180 std::cout << "Testing MTK 1D Laplacian" << std::endl;
00181
00182 TestDefaultConstructorFactoryMethodDefault();
00183 TestDefaultConstructorFactoryMethodFourthOrder();
00184 TestDefaultConstructorFactoryMethodSixthOrder();
00185 TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00186 TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00187 TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold();

```

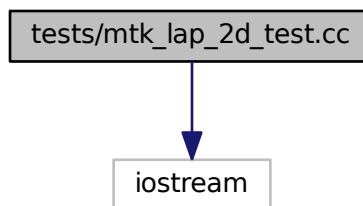
```
00188 TestReturnAsDenseMatrix();
00189 }
00190
00191 #else
00192 #include <iostream>
00193 int main () {
00194 std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00195 std::cout << "Exiting..." << std::endl;
00196 }
00197 #endif
```

## 18.143 tests/mtk\_lap\_2d\_test.cc File Reference

Test file for the [mtk::Lap2D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_lap_2d_test.cc`:



### Functions

- `int main ()`

#### 18.143.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_2d\\_test.cc](#).

#### 18.143.2 Function Documentation

##### 18.143.2.1 `int main ( )`

Definition at line [139](#) of file [mtk\\_lap\\_2d\\_test.cc](#).

## 18.144 mtk\_lap\_2d\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::Lap2D ll;
00068
00069 mtk::Real aa = 0.0;
00070 mtk::Real bb = 1.0;
00071 mtk::Real cc = 0.0;
00072 mtk::Real dd = 1.0;
00073
00074 int nn = 5;
00075 int mm = 5;
00076
00077 mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00078
00079 bool assertion = ll.ConstructLap2D(llg);
00080
00081 if (!assertion) {
00082 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00083 }
00084

```

```

00085 mtk::Tools::EndUnitTestNo(1);
00086 mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091 mtk::Tools::BeginUnitTestNo(2);
00092
00093 mtk::Lap2D ll;
00094
00095 mtk::Real aa = 0.0;
00096 mtk::Real bb = 1.0;
00097 mtk::Real cc = 0.0;
00098 mtk::Real dd = 1.0;
00099
00100 int nn = 5;
00101 int mm = 5;
00102
00103 mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00104
00105 bool assertion = ll.ConstructLap2D(llg);
00106
00107 if (!assertion) {
00108 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00109 }
00110
00111 mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00112
00113 assertion = assertion && (llm.num_rows() != 0);
00114
00115 std::cout << llm << std::endl;
00116
00117 assertion = assertion && llm.WriteToFile("mtk_lap_2d_test_02.dat");
00118
00119 if (!assertion) {
00120 std::cerr << "Error writing to file." << std::endl;
00121 }
00122
00123 mtk::Tools::EndUnitTestNo(2);
00124 mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129 std::cout << "Testing mtk::Lap2D class." << std::endl;
00130
00131 TestDefaultConstructorFactory();
00132 TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140 cout << "This code HAS to be compiled with support for C++11." << endl;
00141 cout << "Exiting..." << endl;
00142 }
00143 #endif

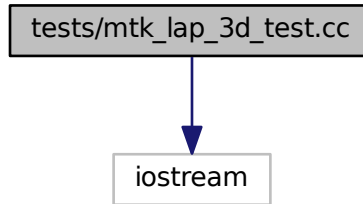
```

## 18.145 tests/mtk\_lap\_3d\_test.cc File Reference

Test file for the [mtk::Lap3D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_lap_3d_test.cc`:



## Functions

- `int` [main](#) ()

### 18.145.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_lap\\_3d\\_test.cc](#).

### 18.145.2 Function Documentation

#### 18.145.2.1 `int` `main` ( )

Definition at line [145](#) of file [mtk\\_lap\\_3d\\_test.cc](#).

## 18.146 `mtk_lap_3d_test.cc`

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```



```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <cmath>
00056 #include <ctime>
00057 #include <iostream>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactory() {
00061 mtk::Tools::BeginUnitTestNo(1);
00062
00063 mtk::Lap3D ll;
00064
00065 mtk::Real aa = 0.0;
00066 mtk::Real bb = 1.0;
00067 mtk::Real cc = 0.0;
00068 mtk::Real dd = 1.0;
00069 mtk::Real ee = 0.0;
00070 mtk::Real ff = 1.0;
00071
00072 int nn = 5;
00073 int mm = 5;
00074 int oo = 5;
00075
00076 mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00077
00078 bool assertion = ll.ConstructLap3D(llg);
00079
00080 if (!assertion) {
00081 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00082 }
00083
00084 mtk::Tools::EndUnitTestNo(1);
00085 mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestReturnAsDenseMatrixWriteToFile() {
00089 mtk::Tools::BeginUnitTestNo(2);
00090
00091 mtk::Lap3D ll;
00092
00093 mtk::Real aa = 0.0;
00094 mtk::Real bb = 1.0;
00095 mtk::Real cc = 0.0;
00096 mtk::Real dd = 1.0;
00097 mtk::Real ee = 0.0;
00098 mtk::Real ff = 1.0;
00099
00100 int nn = 5;

```

```

00106 int mm = 5;
00107 int oo = 5;
00108
00109 mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00110
00111 bool assertion = ll.ConstructLap3D(llg);
00112
00113 if (!assertion) {
00114 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00115 }
00116
00117 mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00118
00119 assertion = assertion && (llm.num_rows() != 0);
00120
00121 std::cout << llm << std::endl;
00122
00123 assertion = assertion && llm.WriteToFile("mtk_lap_3d_test_02.dat");
00124
00125 if (!assertion) {
00126 std::cerr << "Error writing to file." << std::endl;
00127 }
00128
00129 mtk::Tools::EndUnitTestNo(2);
00130 mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135 std::cout << "Testing mtk::Lap3D class." << std::endl;
00136
00137 TestDefaultConstructorFactory();
00138 TestReturnAsDenseMatrixWriteToFile();
00139 }
00140
00141 #else
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146 cout << "This code HAS to be compiled with support for C++11." << endl;
00147 cout << "Exiting..." << endl;
00148 }
00149 #endif

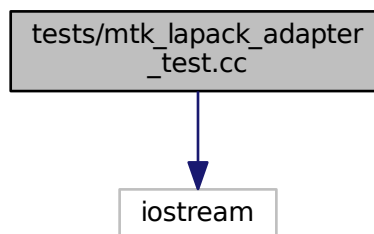
```

## 18.147 tests/mtk\_lapack\_adapter\_test.cc File Reference

Test file for the [mtk::LAPACKAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for mtk\_lapack\_adapter\_test.cc:



## Functions

- `int main ()`

### 18.147.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

**Todo** Test the `mtk::LAPACKAdapter` class.

Definition in file `mtk_lapack_adapter_test.cc`.

### 18.147.2 Function Documentation

#### 18.147.2.1 `int main ( )`

Definition at line 81 of file `mtk_lapack_adapter_test.cc`.

## 18.148 mtk\_lapack\_adapter\_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

```

00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072 std::cout << "Testing mtk::LAPACKAdapter class." << std::endl;
00073
00074 Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082 cout << "This code HAS to be compiled with support for C++11." << endl;
00083 cout << "Exiting..." << endl;
00084 }
00085 #endif

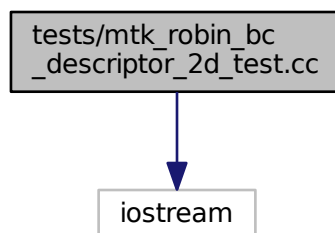
```

## 18.149 tests/mtk\_robin\_bc\_descriptor\_2d\_test.cc File Reference

Test file for the [mtk::RobinBCDescriptor2D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_robin_bc_descriptor_2d_test.cc`:



## Functions

- `int main ()`

## 18.149.1 Detailed Description

### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_robin\\_bc\\_descriptor\\_2d\\_test.cc](#).

## 18.149.2 Function Documentation

### 18.149.2.1 int main ( )

Definition at line 198 of file [mtk\\_robin\\_bc\\_descriptor\\_2d\\_test.cc](#).

## 18.150 mtk\_robin\_bc\_descriptor\_2d\_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
```

```

00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorGetters() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::RobinBCDescriptor2D bcd;
00068
00069 bool assertion{true};
00070
00071 assertion = assertion && bcd.highest_order_diff_west() == -1;
00072 assertion = assertion && bcd.highest_order_diff_east() == -1;
00073 assertion = assertion && bcd.highest_order_diff_south() == -1;
00074 assertion = assertion && bcd.highest_order_diff_north() == -1;
00075
00076 mtk::Tools::EndUnitTestNo(1);
00077 mtk::Tools::Assert(assertion);
00078 }
00079
00080 mtk::Real cc(const mtk::Real &xx, const mtk::Real &yy) {
00081
00082 return mtk::kOne;
00083 }
00084
00085 void TestPushBackImposeOnLaplacianMatrix() {
00086
00087 mtk::Tools::BeginUnitTestNo(2);
00088
00089 mtk::RobinBCDescriptor2D bcd;
00090
00091 bool assertion{true};
00092
00093 bcd.PushBackWestCoeff(cc);
00094 bcd.PushBackEastCoeff(cc);
00095 bcd.PushBackSouthCoeff(cc);
00096 bcd.PushBackNorthCoeff(cc);
00097
00098 assertion = assertion && bcd.highest_order_diff_west() == 0;
00099 assertion = assertion && bcd.highest_order_diff_east() == 0;
00100 assertion = assertion && bcd.highest_order_diff_south() == 0;
00101 assertion = assertion && bcd.highest_order_diff_north() == 0;
00102
00103 mtk::Real aa = 0.0;
00104 mtk::Real bb = 1.0;
00105 mtk::Real cc = 0.0;
00106 mtk::Real dd = 1.0;
00107
00108 int nn = 5;
00109 int mm = 5;
00110
00111 mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00112
00113 mtk::Lap2D ll;
00114
00115 assertion = ll.ConstructLap2D(llg);
00116
00117 if (!assertion) {
00118 std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00119 }
00120
00121 mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00122
00123 assertion = assertion && (llm.num_rows() != 0);
00124
00125 bcd.ImposeOnLaplacianMatrix(ll, llg, llm);
00126
00127 assertion = assertion &&
00128 llm.WriteToFile("mtk_robin_bc_descriptor_2d_test_02.dat");
00129
00130 mtk::Tools::EndUnitTestNo(2);
00131 mtk::Tools::Assert(assertion);
00132 }
00133
00134 mtk::Real ScalarField(const mtk::Real &xx, const mtk::Real &yy) {
00135
00136 mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
00137
00138 return xx*yy*exp(aux);
00139 }
00140
00141 mtk::Real HomogeneousDiricheletBC(const mtk::Real &xx,

```

```

00142 const mtk::Real &tt) {
00143
00144 return mtk::kZero;
00145 }
00146
00147 void TestImposeOnGrid() {
00148 mtk::Tools::BeginUnitTestNo(3);
00149
00150 mtk::Real aa = 0.0;
00151 mtk::Real bb = 1.0;
00152 mtk::Real cc = 0.0;
00153 mtk::Real dd = 1.0;
00154
00155 int nn = 5;
00156 int mm = 5;
00157
00158 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00159
00160 gg.BindScalarField(ScalarField);
00161
00162 mtk::RobinBCDescriptor2D desc;
00163
00164 desc.set_west_condition(HomogeneousDiricheletBC);
00165 desc.set_east_condition(HomogeneousDiricheletBC);
00166 desc.set_south_condition(HomogeneousDiricheletBC);
00167 desc.set_north_condition(HomogeneousDiricheletBC);
00168
00169 desc.ImposeOnGrid(gg);
00170
00171 bool assertion{gg.WriteToFile("mtk_robin_bc_descriptor_2d_test_03.dat",
00172 "x",
00173 "y",
00174 "u(x,y)");};
00175
00176 if(!assertion) {
00177 std::cerr << "Error writing to file." << std::endl;
00178 }
00179
00180 mtk::Tools::EndUnitTestNo(3);
00181 mtk::Tools::Assert(assertion);
00182 }
00183
00184 int main () {
00185
00186 std::cout << "Testing mtk::RobinBCDescriptor2D class." << std::endl;
00187
00188 TestDefaultConstructorGetters();
00189 TestPushBackImposeOnLaplacianMatrix();
00190 TestImposeOnGrid();
00191 }
00192
00193 #else
00194 #include <iostream>
00195 using std::cout;
00196 using std::endl;
00197
00198 int main () {
00199 cout << "This code HAS to be compiled with support for C++11." << endl;
00200 cout << "Exiting..." << endl;
00201 }
00202 #endif

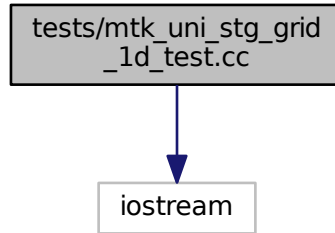
```

## 18.151 tests/mtk\_uni\_stg\_grid\_1d\_test.cc File Reference

Test file for the `mtk::UniStgGrid1D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_uni_stg_grid_1d_test.cc`:



## Functions

- int [main](#) ()

### 18.151.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)

Definition in file [mtk\\_uni\\_stg\\_grid\\_1d\\_test.cc](#).

### 18.151.2 Function Documentation

#### 18.151.2.1 int main ( )

Definition at line [172](#) of file [mtk\\_uni\\_stg\\_grid\\_1d\\_test.cc](#).

## 18.152 mtk\_uni\_stg\_grid\_1d\_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023

```



```

00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062
00063 mtk::Tools::BeginUnitTestNo(1);
00064
00065 mtk::UniStgGrid1D gg;
00066
00067 mtk::Tools::EndUnitTestNo(1);
00068 mtk::Tools::Assert(gg.delta_x() == mtk::kZero);
00069 }
00070
00071 mtk::Real ScalarField(const mtk::Real &xx) {
00072
00073 return 2.0*xx;
00074 }
00075
00076 void TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField() {
00077
00078 mtk::Tools::BeginUnitTestNo(2);
00079
00080 mtk::Real aa = 0.0;
00081 mtk::Real bb = 1.0;
00082
00083 int nn = 5;
00084
00085 mtk::UniStgGrid1D gg(aa, bb, nn);
00086
00087 gg.BindScalarField(ScalarField);
00088
00089 std::cout << gg << std::endl;
00090
00091 mtk::Tools::EndUnitTestNo(2);
00092 mtk::Tools::Assert(gg.delta_x() == 0.2 && gg.
num_cells_x() == 5);
00093 }
00094
00095 void TestBindScalarFieldWriteToFile() {
00096
00097 mtk::Tools::BeginUnitTestNo(3);
00098
00099 mtk::Real aa = 0.0;
00100 mtk::Real bb = 1.0;
00101
00102 int nn = 5;
00103

```

```

00104 mtk::UniStgGrid1D gg(aa, bb, nn);
00105
00106 bool assertion{true};
00107
00108 gg.BindScalarField(ScalarField);
00109
00110 assertion =
00111 assertion &&
00112 gg.discrete_field()[0] == 0.0 &&
00113 gg.discrete_field()[gg.num_cells_x() + 2 - 1] == 2.0;
00114
00115 if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_03.dat", "x", "u(x)")) {
00116 std::cerr << "Error writing to file." << std::endl;
00117 assertion = false;
00118 }
00119
00120 mtk::Tools::EndUnitTestNo(3);
00121 mtk::Tools::Assert(assertion);
00122 }
00123
00124 mtk::Real VectorFieldPComponent(mtk::Real xx) {
00125 return xx*xx;
00126 }
00127
00128 void TestBindVectorField() {
00129
00130 mtk::Tools::BeginUnitTestNo(4);
00131
00132 mtk::Real aa = 0.0;
00133 mtk::Real bb = 1.0;
00134
00135 int nn = 20;
00136
00137 mtk::UniStgGrid1D gg(aa, bb, nn, mtk::VECTOR);
00138
00139 bool assertion{true};
00140
00141 gg.BindVectorField(VectorFieldPComponent);
00142
00143 assertion =
00144 assertion &&
00145 gg.discrete_field()[0] == 0.0 &&
00146 gg.discrete_field()[gg.num_cells_x() + 1 - 1] == 1.0;
00147
00148 if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_04.dat", "x", "v(x)")) {
00149 std::cerr << "Error writing to file." << std::endl;
00150 assertion = false;
00151 }
00152
00153 mtk::Tools::EndUnitTestNo(4);
00154 mtk::Tools::Assert(assertion);
00155 }
00156
00157 int main () {
00158 std::cout << "Testing mtk::UniStgGrid1D class." << std::endl;
00159
00160 TestDefaultConstructor();
00161 TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField();
00162 TestBindScalarFieldWriteToFile();
00163 TestBindVectorField();
00164 }
00165
00166 #else
00167 #include <iostream>
00168 using std::cout;
00169 using std::endl;
00170 int main () {
00171 cout << "This code HAS to be compiled with support for C++11." << endl;
00172 cout << "Exiting..." << endl;
00173 }
00174 #endif

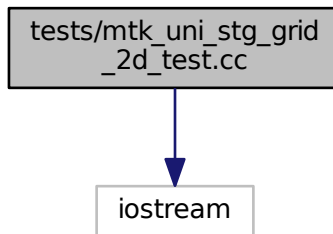
```

## 18.153 tests/mtk\_uni\_stg\_grid\_2d\_test.cc File Reference

Test file for the [mtk::UniStgGrid2D](#) class.

```
#include <iostream>
```

Include dependency graph for mtk\_uni\_stg\_grid\_2d\_test.cc:



### Functions

- int [main](#) ()

#### 18.153.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_2d\\_test.cc](#).

#### 18.153.2 Function Documentation

##### 18.153.2.1 int main ( )

Definition at line [202](#) of file [mtk\\_uni\\_stg\\_grid\\_2d\\_test.cc](#).

## 18.154 mtk\_uni\_stg\_grid\_2d\_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
```

```

00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructor() {
00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::UniStgGrid2D gg;
00068
00069 mtk::Tools::EndUnitTestNo(1);
00070 mtk::Tools::Assert(gg.delta_x() == mtk::kZero && gg.
00071 delta_y() == mtk::kZero);
00072 }
00073
00074 void
00075 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator() {
00076
00077 mtk::Tools::BeginUnitTestNo(2);
00078
00079 mtk::Real aa = 0.0;
00080 mtk::Real bb = 1.0;
00081 mtk::Real cc = 0.0;
00082 mtk::Real dd = 1.0;
00083
00084 int nn = 5;
00085 int mm = 7;
00086
00087 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00088
00089 std::cout << gg << std::endl;
00090
00091 mtk::Tools::EndUnitTestNo(2);
00092 mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00093 abs(gg.delta_y() - 0.142857) <
00094 mtk::kDefaultTolerance);
00095 }
00096
00097 void TestGetters() {
00098

```

```

00097 mtk::Tools::BeginUnitTestNo(3);
00098
00099 mtk::Real aa = 0.0;
00100 mtk::Real bb = 1.0;
00101 mtk::Real cc = 0.0;
00102 mtk::Real dd = 1.0;
00103
00104 int nn = 5;
00105 int mm = 7;
00106
00107 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00108
00109 bool assertion{true};
00110
00111 assertion = assertion && (gg.west_bndy() == aa);
00112 assertion = assertion && (gg.east_bndy() == bb);
00113 assertion = assertion && (gg.num_cells_x() == nn);
00114 assertion = assertion && (gg.south_bndy() == cc);
00115 assertion = assertion && (gg.north_bndy() == dd);
00116 assertion = assertion && (gg.num_cells_y() == mm);
00117
00118 mtk::Tools::EndUnitTestNo(3);
00119 mtk::Tools::Assert(assertion);
00120 }
00121
00122 mtk::Real ScalarField(const mtk::Real &xx, const mtk::Real &yy) {
00123
00124 mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
00125
00126 return xx*yy*exp(aux);
00127 }
00128
00129 void TestBindScalarFieldWriteToFile() {
00130
00131 mtk::Tools::BeginUnitTestNo(4);
00132
00133 mtk::Real aa = 0.0;
00134 mtk::Real bb = 1.0;
00135 mtk::Real cc = 0.0;
00136 mtk::Real dd = 1.0;
00137
00138 int nn = 5;
00139 int mm = 5;
00140
00141 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00142
00143 gg.BindScalarField(ScalarField);
00144
00145 if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_04.dat", "x", "y", "u(x,y)")) {
00146 std::cerr << "Error writing to file." << std::endl;
00147 }
00148
00149 mtk::Tools::EndUnitTestNo(4);
00150 }
00151
00152 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const
mtk::Real &yy) {
00153
00154 return xx + 0.01;
00155 }
00156
00157 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
mtk::Real &yy) {
00158
00159 return yy + 0.01;
00160 }
00161
00162 void TestBindVectorField() {
00163
00164 mtk::Tools::BeginUnitTestNo(5);
00165
00166 mtk::Real aa = 0.0;
00167 mtk::Real bb = 1.0;
00168 mtk::Real cc = 0.0;
00169 mtk::Real dd = 1.0;
00170
00171 int nn = 5;
00172 int mm = 5;
00173
00174 mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00175

```

```

00176 gg.BindVectorField(VectorFieldPComponent, VectorFieldQComponent);
00177
00178 std::cout << gg << std::endl;
00179
00180 if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_05.dat", "x", "y", "v(x,y)")) {
00181 std::cerr << "Error writing to file." << std::endl;
00182 }
00183
00184 mtk::Tools::EndUnitTestNo(5);
00185 }
00186
00187 int main () {
00188 std::cout << "Testing mtk::UniStgGrid2D class." << std::endl;
00189
00190 TestDefaultConstructor();
00191 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator();
00192 TestGetters();
00193 TestBindScalarFieldWriteToFile();
00194 TestBindVectorField();
00195 }
00196
00197 #else
00198 #include <iostream>
00199 using std::cout;
00200 using std::endl;
00202 int main () {
00203 cout << "This code HAS to be compiled with support for C++11." << endl;
00204 cout << "Exiting..." << endl;
00205 }
00206 #endif

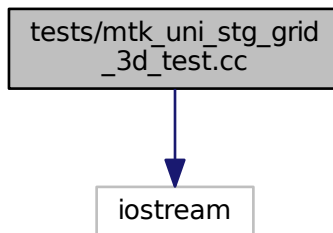
```

## 18.155 tests/mtk\_uni\_stg\_grid\_3d\_test.cc File Reference

Test file for the `mtk::UniStgGrid3D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_uni_stg_grid_3d_test.cc`:



### Functions

- `int main ()`

#### 18.155.1 Detailed Description

**Author**

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk\\_uni\\_stg\\_grid\\_3d\\_test.cc](#).

**18.155.2 Function Documentation****18.155.2.1 int main ( )**

Definition at line 184 of file [mtk\\_uni\\_stg\\_grid\\_3d\\_test.cc](#).

**18.156 mtk\_uni\_stg\_grid\_3d\_test.cc**

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructor() {
```

```

00064
00065 mtk::Tools::BeginUnitTestNo(1);
00066
00067 mtk::UniStgGrid3D gg;
00068
00069 mtk::Tools::EndUnitTestNo(1);
00070 mtk::Tools::Assert(gg.delta_x() == mtk::kZero &&
00071 gg.delta_y() == mtk::kZero &&
00072 gg.delta_z() == mtk::kZero);
00073 }
00074
00075 void
00076 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator() {
00077
00078 mtk::Tools::BeginUnitTestNo(2);
00079
00080 mtk::Real aa = 0.0;
00081 mtk::Real bb = 1.0;
00082 mtk::Real cc = 0.0;
00083 mtk::Real dd = 1.0;
00084 mtk::Real ee = 0.0;
00085 mtk::Real ff = 1.0;
00086
00087 int nn = 5;
00088 int mm = 7;
00089 int oo = 7;
00090
00091 mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00092
00093 std::cout << gg << std::endl;
00094
00095 mtk::Tools::EndUnitTestNo(2);
00096 mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00097 abs(gg.delta_y() - 0.142857) <
00098 mtk::kDefaultTolerance);
00099 }
00100 void TestGetters() {
00101
00102 mtk::Tools::BeginUnitTestNo(3);
00103
00104 mtk::Real aa = 0.0;
00105 mtk::Real bb = 1.0;
00106 mtk::Real cc = 0.0;
00107 mtk::Real dd = 1.0;
00108 mtk::Real ee = 0.0;
00109 mtk::Real ff = 1.0;
00110
00111 int nn = 5;
00112 int mm = 7;
00113 int oo = 6;
00114
00115 mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00116
00117 bool assertion{true};
00118
00119 assertion = assertion && (gg.west_bndy() == aa);
00120 assertion = assertion && (gg.east_bndy() == bb);
00121 assertion = assertion && (gg.num_cells_x() == nn);
00122 assertion = assertion && (gg.south_bndy() == cc);
00123 assertion = assertion && (gg.north_bndy() == dd);
00124 assertion = assertion && (gg.num_cells_y() == mm);
00125 assertion = assertion && (gg.bottom_bndy() == ee);
00126 assertion = assertion && (gg.top_bndy() == ff);
00127 assertion = assertion && (gg.num_cells_z() == oo);
00128
00129 mtk::Tools::EndUnitTestNo(3);
00130 mtk::Tools::Assert(assertion);
00131 }
00132
00133 mtk::Real ScalarField(const mtk::Real &xx,
00134 const mtk::Real &yy,
00135 const mtk::Real &zz) {
00136
00137 return xx + yy + zz;
00138 }
00139
00140 void TestBindScalarFieldWriteToFile() {
00141
00142 mtk::Tools::BeginUnitTestNo(4);
00143

```



```
00144 mtk::Real aa = 0.0;
00145 mtk::Real bb = 1.0;
00146 mtk::Real cc = 0.0;
00147 mtk::Real dd = 1.0;
00148 mtk::Real ee = 0.0;
00149 mtk::Real ff = 1.0;
00150
00151 int nn = 50;
00152 int mm = 50;
00153 int oo = 50;
00154
00155 mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00156
00157 gg.BindScalarField(ScalarField);
00158
00159 if(!gg.WriteToFile("mtk_uni_stg_grid_3d_test_04.dat",
00160 "x",
00161 "y",
00162 "z",
00163 "u(x,y,z)")) {
00164 std::cerr << "Error writing to file." << std::endl;
00165 }
00166
00167 mtk::Tools::EndUnitTestNo(4);
00168 }
00169
00170 int main () {
00171
00172 std::cout << "Testing mtk::UniStgGrid3D class." << std::endl;
00173
00174 TestDefaultConstructor();
00175 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator();
00176 TestGetters();
00177 TestBindScalarFieldWriteToFile();
00178 }
00179
00180 #else
00181 #include <iostream>
00182 using std::cout;
00183 using std::endl;
00184 int main () {
00185 cout << "This code HAS to be compiled with support for C++11." << endl;
00186 cout << "Exiting..." << endl;
00187 }
00188 #endif
```

# Index

BANDED  
Enumerations., [37](#)

COL\_MAJOR  
Enumerations., [37](#)

CRS  
Enumerations., [37](#)

DENSE  
Enumerations., [37](#)

Data structures., [39](#)

Enumerations., [36](#)  
BANDED, [37](#)  
COL\_MAJOR, [37](#)  
CRS, [37](#)  
DENSE, [37](#)  
ROW\_MAJOR, [37](#)  
SCALAR, [36](#)  
SCALAR\_TO\_VECTOR, [36](#)  
VECTOR, [36](#)  
VECTOR\_TO\_SCALAR, [36](#)

Execution tools., [38](#)

Grids., [41](#)

Mimetic operators., [42](#)

mtk, [45](#)  
operator<<, [48](#), [49](#)

Numerical methods., [40](#)

operator<<  
mtk, [48](#), [49](#)

ROW\_MAJOR  
Enumerations., [37](#)

Real  
Roots., [34](#)

Roots., [33](#)  
Real, [34](#)

SCALAR  
Enumerations., [36](#)

SCALAR\_TO\_VECTOR  
Enumerations., [36](#)

VECTOR