

MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.8

Tue Nov 17 2015 12:49:59

Contents

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

14 Module Documentation	31
14.1 Roots.	31
14.1.1 Detailed Description	31
14.1.2 Typedef Documentation	32
14.1.2.1 Real	32
14.1.3 Variable Documentation	32
14.1.3.1 kCriticalOrderAccuracyDiv	32
14.1.3.2 kCriticalOrderAccuracyGrad	32
14.1.3.3 kDefaultMimeticThreshold	32
14.1.3.4 kDefaultOrderAccuracy	32
14.1.3.5 kDefaultTolerance	32
14.1.3.6 kOne	32
14.1.3.7 kZero	32
14.2 Enumerations.	33
14.2.1 Detailed Description	33
14.2.2 Enumeration Type Documentation	33
14.2.2.1 DirInterp	33
14.2.2.2 FieldNature	33
14.2.2.3 MatrixOrdering	34
14.2.2.4 MatrixStorage	34
14.3 Execution tools.	35
14.3.1 Detailed Description	35
14.4 Data structures.	36
14.4.1 Detailed Description	36
14.5 Numerical methods.	37
14.5.1 Detailed Description	37
14.6 Grids.	38
14.6.1 Detailed Description	38
14.7 Mimetic operators.	39
14.7.1 Detailed Description	39
15 Namespace Documentation	41
15.1 mtk Namespace Reference	41
15.1.1 Function Documentation	43
15.1.1.1 operator<<	43
15.1.1.2 operator<<	43
15.1.1.3 operator<<	43

15.1.1.4	operator<<	43
15.1.1.5	operator<<	44
15.1.1.6	operator<<	44
15.1.1.7	operator<<	44
15.1.1.8	saxpy_	45
15.1.1.9	sgels_	45
15.1.1.10	sgemm_	46
15.1.1.11	sgemv_	47
15.1.1.12	sgeqrf_	47
15.1.1.13	sgesv_	47
15.1.1.14	snrm2_	48
15.1.1.15	sormqr_	48
16	Class Documentation	51
16.1	mtk::BCDesc1D Class Reference	51
16.1.1	Detailed Description	51
16.1.2	Member Function Documentation	51
16.1.2.1	ImposeOnGrid	52
16.1.2.2	ImposeOnOperatorMatrix	53
16.2	mtk::BCDesc2D Class Reference	54
16.2.1	Detailed Description	54
16.2.2	Member Function Documentation	55
16.2.2.1	ImposeOnGrid	55
16.2.2.2	ImposeOnOperatorMatrix	55
16.3	mtk::BLASAdapter Class Reference	56
16.3.1	Detailed Description	57
16.3.2	Member Function Documentation	57
16.3.2.1	RealAXPY	57
16.3.2.2	RealDenseMM	58
16.3.2.3	RealDenseMV	59
16.3.2.4	RealNRM2	61
16.3.2.5	RelNorm2Error	62
16.4	mtk::DenseMatrix Class Reference	62
16.4.1	Detailed Description	65
16.4.2	Constructor & Destructor Documentation	65
16.4.2.1	DenseMatrix	65
16.4.2.2	DenseMatrix	65

16.4.2.3	DenseMatrix	66
16.4.2.4	DenseMatrix	67
16.4.2.5	DenseMatrix	67
16.4.2.6	~DenseMatrix	68
16.4.3	Member Function Documentation	68
16.4.3.1	data	68
16.4.3.2	GetValue	69
16.4.3.3	Kron	70
16.4.3.4	matrix_properties	71
16.4.3.5	num_cols	72
16.4.3.6	num_rows	72
16.4.3.7	operator=	73
16.4.3.8	operator==	74
16.4.3.9	OrderColMajor	75
16.4.3.10	OrderRowMajor	75
16.4.3.11	SetOrdering	76
16.4.3.12	SetValue	77
16.4.3.13	Transpose	78
16.4.3.14	WriteToFile	78
16.4.4	Friends And Related Function Documentation	79
16.4.4.1	operator<<	79
16.4.5	Member Data Documentation	79
16.4.5.1	data_	79
16.4.5.2	matrix_properties_	79
16.5	mtk::Div1D Class Reference	79
16.5.1	Detailed Description	82
16.5.2	Constructor & Destructor Documentation	82
16.5.2.1	Div1D	82
16.5.2.2	Div1D	82
16.5.2.3	~Div1D	82
16.5.3	Member Function Documentation	83
16.5.3.1	AssembleOperator	83
16.5.3.2	coeffs_interior	83
16.5.3.3	ComputePreliminaryApproximations	83
16.5.3.4	ComputeRationalBasisNullSpace	84
16.5.3.5	ComputeStencilBoundaryGrid	85
16.5.3.6	ComputeStencilInteriorGrid	85

16.5.3.7	ComputeWeights	86
16.5.3.8	ConstructDiv1D	87
16.5.3.9	mim_bndy	87
16.5.3.10	num_bndy_coeffs	88
16.5.3.11	ReturnAsDenseMatrix	88
16.5.3.12	weights_cbs	89
16.5.3.13	weights_crs	89
16.5.4	Friends And Related Function Documentation	89
16.5.4.1	operator<<	89
16.5.5	Member Data Documentation	89
16.5.5.1	coeffs_interior_	89
16.5.5.2	dim_null_	89
16.5.5.3	divergence_	90
16.5.5.4	divergence_length_	90
16.5.5.5	mim_bndy_	90
16.5.5.6	mimetic_threshold_	90
16.5.5.7	minrow_	90
16.5.5.8	num_bndy_coeffs_	90
16.5.5.9	order_accuracy_	90
16.5.5.10	prem_apps_	90
16.5.5.11	rat_basis_null_space_	90
16.5.5.12	row_	90
16.5.5.13	weights_cbs_	90
16.5.5.14	weights_crs_	91
16.6	mtk::Div2D Class Reference	91
16.6.1	Detailed Description	93
16.6.2	Constructor & Destructor Documentation	93
16.6.2.1	Div2D	93
16.6.2.2	Div2D	93
16.6.2.3	~Div2D	93
16.6.3	Member Function Documentation	93
16.6.3.1	ConstructDiv2D	93
16.6.3.2	ReturnAsDenseMatrix	94
16.6.4	Member Data Documentation	95
16.6.4.1	divergence_	95
16.6.4.2	mimetic_threshold_	95
16.6.4.3	order_accuracy_	95

16.7	mtk::GLPKAdapter Class Reference	95
16.7.1	Detailed Description	96
16.7.2	Member Function Documentation	96
16.7.2.1	SolveSimplexAndCompare	96
16.8	mtk::Grad1D Class Reference	98
16.8.1	Detailed Description	101
16.8.2	Constructor & Destructor Documentation	101
16.8.2.1	Grad1D	101
16.8.2.2	Grad1D	101
16.8.2.3	~Grad1D	102
16.8.3	Member Function Documentation	102
16.8.3.1	AssembleOperator	102
16.8.3.2	coeffs_interior	102
16.8.3.3	ComputePreliminaryApproximations	102
16.8.3.4	ComputeRationalBasisNullSpace	103
16.8.3.5	ComputeStencilBoundaryGrid	104
16.8.3.6	ComputeStencilInteriorGrid	104
16.8.3.7	ComputeWeights	105
16.8.3.8	ConstructGrad1D	105
16.8.3.9	mim_bndy	106
16.8.3.10	num_bndy_coeffs	107
16.8.3.11	ReturnAsDenseMatrix	107
16.8.3.12	ReturnAsDenseMatrix	107
16.8.3.13	ReturnAsDimensionlessDenseMatrix	108
16.8.3.14	weights_cbs	108
16.8.3.15	weights_crs	109
16.8.4	Friends And Related Function Documentation	109
16.8.4.1	operator<<	109
16.8.5	Member Data Documentation	109
16.8.5.1	coeffs_interior_	109
16.8.5.2	dim_null_	109
16.8.5.3	gradient_	109
16.8.5.4	gradient_length_	109
16.8.5.5	mim_bndy_	109
16.8.5.6	mimetic_threshold_	109
16.8.5.7	minrow_	110
16.8.5.8	num_bndy_approxs_	110

16.8.5.9 num_bndy_coeffs_	110
16.8.5.10 order_accuracy_	110
16.8.5.11 prem_apps_	110
16.8.5.12 rat_basis_null_space_	110
16.8.5.13 row_	110
16.8.5.14 weights_cbs_	110
16.8.5.15 weights_crs_	110
16.9 mtk::Grad2D Class Reference	110
16.9.1 Detailed Description	112
16.9.2 Constructor & Destructor Documentation	112
16.9.2.1 Grad2D	112
16.9.2.2 Grad2D	112
16.9.2.3 ~Grad2D	112
16.9.3 Member Function Documentation	112
16.9.3.1 ConstructGrad2D	112
16.9.3.2 ReturnAsDenseMatrix	113
16.9.4 Member Data Documentation	114
16.9.4.1 gradient_	114
16.9.4.2 mimetic_threshold_	114
16.9.4.3 order_accuracy_	114
16.10 mtk::Interp1D Class Reference	114
16.10.1 Detailed Description	116
16.10.2 Constructor & Destructor Documentation	116
16.10.2.1 Interp1D	116
16.10.2.2 Interp1D	116
16.10.2.3 ~Interp1D	116
16.10.3 Member Function Documentation	116
16.10.3.1 coeffs_interior	116
16.10.3.2 ConstructInterp1D	116
16.10.3.3 ReturnAsDenseMatrix	117
16.10.4 Friends And Related Function Documentation	117
16.10.4.1 operator<<	117
16.10.5 Member Data Documentation	118
16.10.5.1 coeffs_interior_	118
16.10.5.2 dir_interp_	118
16.10.5.3 order_accuracy_	118
16.11 mtk::Interp2D Class Reference	118

16.11.1 Detailed Description	120
16.11.2 Constructor & Destructor Documentation	120
16.11.2.1 Interp2D	120
16.11.2.2 Interp2D	120
16.11.2.3 ~Interp2D	120
16.11.3 Member Function Documentation	120
16.11.3.1 ConstructInterp2D	120
16.11.3.2 ReturnAsDenseMatrix	121
16.11.4 Member Data Documentation	121
16.11.4.1 interpolator_	121
16.11.4.2 mimetic_threshold_	121
16.11.4.3 order_accuracy_	121
16.12mtk::Lap1D Class Reference	121
16.12.1 Detailed Description	122
16.12.2 Constructor & Destructor Documentation	122
16.12.2.1 Lap1D	122
16.12.2.2 Lap1D	122
16.12.2.3 ~Lap1D	123
16.12.3 Member Function Documentation	123
16.12.3.1 ConstructLap1D	123
16.12.3.2 data	124
16.12.3.3 ReturnAsDenseMatrix	125
16.12.4 Friends And Related Function Documentation	125
16.12.4.1 operator<<	125
16.12.5 Member Data Documentation	125
16.12.5.1 laplacian_	125
16.12.5.2 laplacian_length_	126
16.12.5.3 mimetic_threshold_	126
16.12.5.4 order_accuracy_	126
16.13mtk::Lap2D Class Reference	126
16.13.1 Detailed Description	128
16.13.2 Constructor & Destructor Documentation	128
16.13.2.1 Lap2D	128
16.13.2.2 Lap2D	128
16.13.2.3 ~Lap2D	128
16.13.3 Member Function Documentation	129
16.13.3.1 ConstructLap2D	129

16.13.3.2 data	129
16.13.3.3 ReturnAsDenseMatrix	129
16.13.4 Member Data Documentation	130
16.13.4.1 laplacian_	130
16.13.4.2 mimetic_threshold_	130
16.13.4.3 order_accuracy_	130
16.14mtk::LAPACKAdapter Class Reference	130
16.14.1 Detailed Description	131
16.14.2 Member Function Documentation	131
16.14.2.1 QRFactorDenseMatrix	131
16.14.2.2 SolveDenseSystem	132
16.14.2.3 SolveDenseSystem	133
16.14.2.4 SolveDenseSystem	134
16.14.2.5 SolveRectangularDenseSystem	135
16.15mtk::Matrix Class Reference	136
16.15.1 Detailed Description	139
16.15.2 Constructor & Destructor Documentation	139
16.15.2.1 Matrix	139
16.15.2.2 Matrix	140
16.15.2.3 ~Matrix	141
16.15.3 Member Function Documentation	141
16.15.3.1 abs_density	141
16.15.3.2 abs_sparsity	141
16.15.3.3 bandwidth	141
16.15.3.4 IncreaseNumNull	141
16.15.3.5 IncreaseNumZero	142
16.15.3.6 kl	142
16.15.3.7 ku	142
16.15.3.8 ld	142
16.15.3.9 num_cols	142
16.15.3.10num_non_null	143
16.15.3.11num_non_zero	143
16.15.3.12num_null	143
16.15.3.13num_rows	143
16.15.3.14num_values	144
16.15.3.15num_zero	144
16.15.3.16ordering	144

16.15.3.17	rel_density	145
16.15.3.18	rel_sparsity	145
16.15.3.19	set_num_cols	145
16.15.3.20	set_num_null	146
16.15.3.21	set_num_rows	147
16.15.3.22	set_num_zero	147
16.15.3.23	set_ordering	148
16.15.3.24	set_storage	149
16.15.3.25	storage	149
16.15.4	Member Data Documentation	150
16.15.4.1	abs_density_	150
16.15.4.2	abs_sparsity_	150
16.15.4.3	bandwidth_	150
16.15.4.4	kl_	150
16.15.4.5	ku_	150
16.15.4.6	ld_	150
16.15.4.7	num_cols_	150
16.15.4.8	num_non_null_	151
16.15.4.9	num_non_zero_	151
16.15.4.10	num_null_	151
16.15.4.11	num_rows_	151
16.15.4.12	num_values_	151
16.15.4.13	num_zero_	151
16.15.4.14	ordering_	151
16.15.4.15	rel_density_	151
16.15.4.16	rel_sparsity_	151
16.15.4.17	storage_	151
16.16	mtk::Quad1D Class Reference	152
16.16.1	Detailed Description	153
16.16.2	Constructor & Destructor Documentation	153
16.16.2.1	Quad1D	153
16.16.2.2	Quad1D	153
16.16.2.3	~Quad1D	154
16.16.3	Member Function Documentation	154
16.16.3.1	degree_approximation	154
16.16.3.2	Integrate	154
16.16.3.3	weights	154

16.16.4 Friends And Related Function Documentation	154
16.16.4.1 operator<<	154
16.16.5 Member Data Documentation	154
16.16.5.1 degree_approximation_	154
16.16.5.2 weights_	154
16.17mtk::Tools Class Reference	154
16.17.1 Detailed Description	155
16.17.2 Member Function Documentation	155
16.17.2.1 Assert	155
16.17.2.2 BeginUnitTestNo	156
16.17.2.3 EndUnitTestNo	156
16.17.2.4 Prevent	156
16.17.3 Member Data Documentation	158
16.17.3.1 begin_time_	158
16.17.3.2 duration_	159
16.17.3.3 test_number_	159
16.18mtk::UniStgGrid1D Class Reference	159
16.18.1 Detailed Description	162
16.18.2 Constructor & Destructor Documentation	162
16.18.2.1 UniStgGrid1D	162
16.18.2.2 UniStgGrid1D	162
16.18.2.3 UniStgGrid1D	162
16.18.2.4 ~UniStgGrid1D	162
16.18.3 Member Function Documentation	163
16.18.3.1 BindScalarField	163
16.18.3.2 BindVectorField	163
16.18.3.3 delta_x	164
16.18.3.4 discrete_domain_x	164
16.18.3.5 discrete_field_u	164
16.18.3.6 east_bndy_x	165
16.18.3.7 num_cells_x	165
16.18.3.8 west_bndy_x	165
16.18.3.9 WriteToFile	165
16.18.4 Friends And Related Function Documentation	166
16.18.4.1 operator<<	166
16.18.5 Member Data Documentation	166
16.18.5.1 delta_x_	166

16.18.5.2 discrete_domain_x_	166
16.18.5.3 discrete_field_u_	166
16.18.5.4 east_bndy_x_	166
16.18.5.5 nature_	166
16.18.5.6 num_cells_x_	166
16.18.5.7 west_bndy_x_	166
16.19mtk::UniStgGrid2D Class Reference	167
16.19.1 Detailed Description	169
16.19.2 Constructor & Destructor Documentation	170
16.19.2.1 UniStgGrid2D	170
16.19.2.2 UniStgGrid2D	170
16.19.2.3 UniStgGrid2D	170
16.19.2.4 ~UniStgGrid2D	170
16.19.3 Member Function Documentation	171
16.19.3.1 BindScalarField	171
16.19.3.2 BindVectorField	171
16.19.3.3 BindVectorFieldPComponent	172
16.19.3.4 BindVectorFieldQComponent	172
16.19.3.5 delta_x	173
16.19.3.6 delta_y	173
16.19.3.7 discrete_domain_x	173
16.19.3.8 discrete_domain_y	173
16.19.3.9 discrete_field	173
16.19.3.10east_bndy	173
16.19.3.11nature	174
16.19.3.12north_bndy	174
16.19.3.13num_cells_x	174
16.19.3.14num_cells_y	174
16.19.3.15south_bndy	175
16.19.3.16west_bndy	175
16.19.3.17WriteToFile	176
16.19.4 Friends And Related Function Documentation	176
16.19.4.1 operator<<	176
16.19.5 Member Data Documentation	177
16.19.5.1 delta_x_	177
16.19.5.2 delta_y_	177
16.19.5.3 discrete_domain_x_	177

16.19.5.4 discrete_domain_y_	177
16.19.5.5 discrete_field_	177
16.19.5.6 east_bndy_	177
16.19.5.7 nature_	177
16.19.5.8 north_bndy_	177
16.19.5.9 num_cells_x_	177
16.19.5.10 num_cells_y_	177
16.19.5.11 south_bndy_	177
16.19.5.12 west_bndy_	178
17 File Documentation	179
17.1 examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference	179
17.1.1 Detailed Description	179
17.1.2 Function Documentation	180
17.1.2.1 main	180
17.2 minimalistic_poisson_1d.cc	180
17.3 examples/poisson_1d/poisson_1d.cc File Reference	182
17.3.1 Detailed Description	182
17.3.2 Function Documentation	183
17.3.2.1 main	183
17.4 poisson_1d.cc	183
17.5 include/mtk.h File Reference	186
17.5.1 Detailed Description	187
17.6 mtk.h	187
17.7 include/mtk_bc_desc_1d.h File Reference	188
17.7.1 Detailed Description	189
17.8 mtk_bc_desc_1d.h	190
17.9 include/mtk_bc_desc_2d.h File Reference	191
17.9.1 Detailed Description	192
17.10 mtk_bc_desc_2d.h	192
17.11 include/mtk_blas_adapter.h File Reference	193
17.11.1 Detailed Description	194
17.12 mtk_blas_adapter.h	195
17.13 include/mtk_dense_matrix.h File Reference	196
17.13.1 Detailed Description	197
17.14 mtk_dense_matrix.h	197
17.15 include/mtk_div_1d.h File Reference	199

17.15.1 Detailed Description	200
17.16mtk_div_1d.h	200
17.17include/mtk_div_2d.h File Reference	201
17.17.1 Detailed Description	203
17.18mtk_div_2d.h	203
17.19include/mtk_enums.h File Reference	204
17.19.1 Detailed Description	205
17.20mtk_enums.h	205
17.21include/mtk_glpk_adapter.h File Reference	206
17.21.1 Detailed Description	207
17.22mtk_glpk_adapter.h	207
17.23include/mtk_grad_1d.h File Reference	208
17.23.1 Detailed Description	209
17.24mtk_grad_1d.h	210
17.25include/mtk_grad_2d.h File Reference	211
17.25.1 Detailed Description	213
17.26mtk_grad_2d.h	213
17.27include/mtk_interp_1d.h File Reference	214
17.27.1 Detailed Description	215
17.28mtk_interp_1d.h	215
17.29include/mtk_interp_2d.h File Reference	217
17.29.1 Detailed Description	217
17.30mtk_interp_2d.h	218
17.31include/mtk_lap_1d.h File Reference	219
17.31.1 Detailed Description	220
17.32mtk_lap_1d.h	220
17.33include/mtk_lap_2d.h File Reference	221
17.33.1 Detailed Description	223
17.34mtk_lap_2d.h	223
17.35include/mtk_lapack_adapter.h File Reference	224
17.35.1 Detailed Description	225
17.36mtk_lapack_adapter.h	225
17.37include/mtk_matrix.h File Reference	226
17.37.1 Detailed Description	227
17.38mtk_matrix.h	227
17.39include/mtk_quad_1d.h File Reference	229
17.39.1 Detailed Description	230

17.40	mtk_quad_1d.h	231
17.41	include/mtk_roots.h File Reference	232
17.41.1	Detailed Description	233
17.42	mtk_roots.h	233
17.43	include/mtk_tools.h File Reference	234
17.43.1	Detailed Description	235
17.44	mtk_tools.h	235
17.45	include/mtk_uni_stg_grid_1d.h File Reference	236
17.45.1	Detailed Description	237
17.46	mtk_uni_stg_grid_1d.h	237
17.47	include/mtk_uni_stg_grid_2d.h File Reference	239
17.47.1	Detailed Description	240
17.48	mtk_uni_stg_grid_2d.h	240
17.49	Makefile.inc File Reference	242
17.50	Makefile.inc	242
17.51	README.md File Reference	244
17.52	README.md	244
17.53	src/mtk_bc_desc_1d.cc File Reference	246
17.53.1	Detailed Description	246
17.54	mtk_bc_desc_1d.cc	246
17.55	src/mtk_bc_desc_2d.cc File Reference	248
17.55.1	Detailed Description	248
17.56	mtk_bc_desc_2d.cc	248
17.57	src/mtk_blas_adapter.cc File Reference	250
17.57.1	Detailed Description	250
17.58	mtk_blas_adapter.cc	251
17.59	src/mtk_dense_matrix.cc File Reference	254
17.60	mtk_dense_matrix.cc	255
17.61	src/mtk_div_1d.cc File Reference	262
17.61.1	Detailed Description	263
17.62	mtk_div_1d.cc	263
17.63	src/mtk_div_2d.cc File Reference	280
17.63.1	Detailed Description	281
17.64	mtk_div_2d.cc	281
17.65	src/mtk_glpk_adapter.cc File Reference	283
17.65.1	Detailed Description	283
17.66	mtk_glpk_adapter.cc	284

17.67src/mtk_grad_1d.cc File Reference	288
17.67.1 Detailed Description	288
17.68mtk_grad_1d.cc	289
17.69src/mtk_grad_2d.cc File Reference	307
17.69.1 Detailed Description	308
17.70mtk_grad_2d.cc	308
17.71src/mtk_interp_1d.cc File Reference	310
17.71.1 Detailed Description	311
17.72mtk_interp_1d.cc	311
17.73src/mtk_lap_1d.cc File Reference	313
17.73.1 Detailed Description	314
17.74mtk_lap_1d.cc	314
17.75src/mtk_lap_2d.cc File Reference	318
17.75.1 Detailed Description	318
17.76mtk_lap_2d.cc	319
17.77src/mtk_lapack_adapter.cc File Reference	320
17.77.1 Detailed Description	321
17.78mtk_lapack_adapter.cc	322
17.79src/mtk_matrix.cc File Reference	329
17.79.1 Detailed Description	329
17.80mtk_matrix.cc	330
17.81src/mtk_tools.cc File Reference	333
17.81.1 Detailed Description	334
17.82mtk_tools.cc	334
17.83src/mtk_uni_stg_grid_1d.cc File Reference	336
17.83.1 Detailed Description	336
17.84mtk_uni_stg_grid_1d.cc	337
17.85src/mtk_uni_stg_grid_2d.cc File Reference	340
17.85.1 Detailed Description	340
17.86mtk_uni_stg_grid_2d.cc	341
17.87tests/mtk_blas_adapter_test.cc File Reference	346
17.87.1 Detailed Description	347
17.87.2 Function Documentation	347
17.87.2.1 main	347
17.88mtk_blas_adapter_test.cc	347
17.89tests/mtk_dense_matrix_test.cc File Reference	349
17.89.1 Detailed Description	349

17.89.2 Function Documentation	349
17.89.2.1 main	349
17.90mtk_dense_matrix_test.cc	350
17.91tests/mtk_div_1d_test.cc File Reference	354
17.91.1 Detailed Description	354
17.91.2 Function Documentation	354
17.91.2.1 main	354
17.92mtk_div_1d_test.cc	354
17.93tests/mtk_glpk_adapter_test.cc File Reference	358
17.93.1 Detailed Description	359
17.93.2 Function Documentation	359
17.93.2.1 main	359
17.94mtk_glpk_adapter_test.cc	359
17.95tests/mtk_grad_1d_test.cc File Reference	360
17.95.1 Detailed Description	360
17.95.2 Function Documentation	361
17.95.2.1 main	361
17.96mtk_grad_1d_test.cc	361
17.97tests/mtk_grad_2d_test.cc File Reference	365
17.97.1 Detailed Description	365
17.97.2 Function Documentation	365
17.97.2.1 main	365
17.98mtk_grad_2d_test.cc	365
17.99tests/mtk_interp_1d_test.cc File Reference	367
17.99.1 Detailed Description	368
17.99.2 Function Documentation	368
17.99.2.1 main	368
17.100mtk_interp_1d_test.cc	368
17.101tests/mtk_lap_1d_test.cc File Reference	369
17.101.1 Detailed Description	370
17.101.2 Function Documentation	370
17.101.2.1 main	370
17.102mtk_lap_1d_test.cc	370
17.103tests/mtk_lapack_adapter_test.cc File Reference	373
17.103.1 Detailed Description	373
17.103.2 Function Documentation	373
17.103.2.1 main	373

17.104	mtk_lapack_adapter_test.cc	374
17.105	tests/mtk_uni_stg_grid_1d_test.cc File Reference	375
17.105.1	Detailed Description	375
17.105.2	Function Documentation	375
17.105.2.1	main	375
17.106	mtk_uni_stg_grid_1d_test.cc	375
17.107	tests/mtk_uni_stg_grid_2d_test.cc File Reference	378
17.107.1	Detailed Description	378
17.107.2	Function Documentation	378
17.107.2.1	main	378
17.108	mtk_uni_stg_grid_2d_test.cc	378
Index		382

Chapter 1

Introduction

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

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

An older version of this library is available outside of GitHub... just email me about it, and you can have it... it is ugly, yet functional and more complete.

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 Flavors

The MTK collection of wrappers is:

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

Others are being designed and developed.

1.3 Contact, Support and Credits

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

Developers are members of:

1. Mimetic Numerical Methods Research and Development Group.
2. Computational Geoscience Research and Development Group.
3. Ocean Modeling Research and Development Group.

Currently the developers are:

1. **Eduardo J. Sanchez, Ph.D.** - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu) - [ejspeiro](mailto:ejspeiro@mail.sdsu.edu)
2. Jose E. Castillo, Ph.D. - [jcastillo at mail dot sdsu dot edu](mailto:jcastillo@mail.sdsu.edu)
3. Guillermo F. Miranda, Ph.D. - [unigrav at hotmail dot com](mailto:unigrav@hotmail.com)
4. Christopher P. Paolini, Ph.D. - [paolini at engineering dot sdsu dot edu](mailto:paolini@engineering.sdsu.edu)
5. Angel Boada.
6. Johnny Corbino.
7. Raul Vargas–Navarro.

1.4 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. Julia Rossi.

Chapter 2

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. Compiler: gcc version 4.4.5 (Ubuntu/Linaro 4.4.4-14ubuntu5). Copyright (C) 2013 Free Software Foundation, Inc.
3. Debugger: GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1. Copyright (C) 2014 Free Software Foundation, Inc.
4. Memory Profiler: valgrind-3.10.0.SVN.

Chapter 3

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 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 4

Read Me File and Installation Instructions

README File for the Mimetic Methods Toolkit (MTK)

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

1. Description

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

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

An older version of this library is available outside of GitHub... just email me about it, and you can have it... it is ugly, yet functional and more complete.

2. Dependencies

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

`$(HOME)/Libraries/`

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/>
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 examples):

```
----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
```

Examples and tests will also be built.

4. Frequently Asked Questions

Q: Why haven't you guys implemented GBS to build the library?

A: I'm on it as we speak! ;)

Q: Is there any main reference when it comes to the theory on Mimetic Methods?

A: Yes! Check: <http://www.csrc.sdsu.edu/mimetic-book>

Q: Do I need to generate the documentation myself?

A: You can if you want to... but if you DO NOT want to, just go to our website.

5. Contact, Support, and Credits

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

Developers are members of:

1. Mimetic Numerical Methods Research and Development Group.
2. Computational Geoscience Research and Development Group.
3. Ocean Modeling Research and Development Group.

Currently the developers are:

Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu - ejspeiro

2. Jose E. Castillo, Ph.D. - jcastillo at mail dot sdsu dot edu
3. Guillermo F. Miranda, Ph.D. - unigrav at hotmail dot com
4. Christopher P. Paolini, Ph.D. - paolini at engineering dot sdsu dot edu
5. Angel Boada.
6. Johnny Corbino.
7. Raul Vargas-Navarro.

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

Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu - ejspeiro

Thanks and happy coding!

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 examples:

1. Linux 3.2.0-23-generic-pae #36-Ubuntu SMP i386 GNU/Linux.
Intel(R) Pentium(R) M CPU 1.73 GHz 2048 KB of cache and stepping of 8.
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5).
2. Linux 3.13.0-67-generic #110-Ubuntu SMP x86_64 GNU/Linux.
Intel(R) Core(TM) i7-4700MQ CPU 2.40 GHz 6144 KB of cache and stepping of 3.
gcc version 4.8.4 (Ubuntu 4.4.4-2ubuntu1~14.04).
3. Linux 3.16.7-29-desktop #1 SMP PREEMPT (6be6a97) x86_64 GNU/Linux
Intel(R) Core(TM) i7-4600U CPU 2.10 GHz 4096 KB of cache and a stepping of 1.
gcc (SUSE Linux) 4.8.3 20140627 [gcc-4_8-branch revision 212064].

Further architectures will be tested!

Chapter 6

Examples

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

Chapter 7

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 8

Todo List

Member [mtk::DenseMatrix::Kron](#) (const DenseMatrix &aa, const DenseMatrix &bb)

Implement Kronecker product using the BLAS.

Member [mtk::DenseMatrix::OrderColMajor](#) ()

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

Member [mtk::DenseMatrix::OrderRowMajor](#) ()

Improve this so that no new ammayes 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](#) ()

Review the definition of sparse matrices properties.

Member [mtk::Matrix::IncreaseNumZero](#) ()

Review the definition of sparse matrices properties.

Member [mtk::Tools::Prevent](#) (const bool complement, const char *fname, int lineno, const char *fxname)

Check if this is the best way of stalling execution.

Member [mtk::Tools::test_number_](#)

Check usage of static methods and private members.

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](#)

Documentation should (better?) capture effects from selective compilation.

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.

Chapter 9

Bug List

Member `mtk::Matrix::set_num_null` (int in)

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

Member `mtk::Matrix::set_num_zero` (int in)

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

Chapter 10

Module Index

10.1 Modules

Here is a list of all modules:

Roots.	31
Enumerations.	33
Execution tools.	35
Data structures.	36
Numerical methods.	37
Grids.	38
Mimetic operators.	39

Chapter 11

Namespace Index

11.1 Namespace List

Here is a list of all namespaces with brief descriptions:

mtk	Mimetic Methods Toolkit namespace	41
---------------------	---	--------------------

Chapter 12

Class Index

12.1 Class List

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

mtk::BCDesc1D	51
mtk::BCDesc2D	54
mtk::BLASAdapter	
Adapter class for the BLAS API	56
mtk::DenseMatrix	
Defines a common dense matrix, using a 1D array	62
mtk::Div1D	
Implements a 1D mimetic divergence operator	79
mtk::Div2D	91
mtk::GLPKAdapter	
Adapter class for the GLPK API	95
mtk::Grad1D	
Implements a 1D mimetic gradient operator	98
mtk::Grad2D	110
mtk::Interp1D	
Implements a 1D interpolation operator	114
mtk::Interp2D	118
mtk::Lap1D	
Implements a 1D mimetic Laplacian operator	121
mtk::Lap2D	126
mtk::LAPACKAdapter	
Adapter class for the LAPACK API	130
mtk::Matrix	
Definition of the representation of a matrix in the MTK	136
mtk::Quad1D	
Implements a 1D mimetic quadrature	152
mtk::Tools	
Tool manager class	154
mtk::UniStgGrid1D	
Uniform 1D Staggered Grid	159
mtk::UniStgGrid2D	
Uniform 2D Staggered Grid	167

Chapter 13

File Index

13.1 File List

Here is a list of all files with brief descriptions:

Makefile.inc	242
examples/minimalistic_poisson_1d/ minimalistic_poisson_1d.cc	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	179
examples/poisson_1d/ poisson_1d.cc	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	182
include/ mtk.h	
Includes the entire API	186
include/ mtk_bc_desc_1d.h	
Enforces boundary conditions in either the operator or the grid	188
include/ mtk_bc_desc_2d.h	
Enforces boundary conditions in either the operator or the grid	191
include/ mtk_blas_adapter.h	
Adapter class for the BLAS API	193
include/ mtk_dense_matrix.h	
Defines a common dense matrix, using a 1D array	196
include/ mtk_div_1d.h	
Includes the definition of the class Div1D	199
include/ mtk_div_2d.h	
Includes the definition of the class Div2D	201
include/ mtk_enums.h	
Considered enumeration types in the MTK	204
include/ mtk_glpk_adapter.h	
Adapter class for the GLPK API	206
include/ mtk_grad_1d.h	
Includes the definition of the class Grad1D	208
include/ mtk_grad_2d.h	
Includes the definition of the class Grad2D	211
include/ mtk_interp_1d.h	
Includes the definition of the class Interp1D	214
include/ mtk_interp_2d.h	
Includes the definition of the class Interp2D	217
include/ mtk_lap_1d.h	
Includes the definition of the class Lap1D	219

include/mtk_lap_2d.h	Includes the implementation of the class Lap2D	221
include/mtk_lapack_adapter.h	Adapter class for the LAPACK API	224
include/mtk_matrix.h	Definition of the representation of a matrix in the MTK	226
include/mtk_quad_1d.h	Includes the definition of the class Quad1D	229
include/mtk_roots.h	Fundamental definitions to be used across all classes of the MTK	232
include/mtk_tools.h	Tool manager class	234
include/mtk_uni_stg_grid_1d.h	Definition of an 1D uniform staggered grid	236
include/mtk_uni_stg_grid_2d.h	Definition of an 2D uniform staggered grid	239
src/mtk_bc_desc_1d.cc	Enforces boundary conditions in either the operator or the grid	246
src/mtk_bc_desc_2d.cc	Enforces boundary conditions in either the operator or the grid	248
src/mtk_blas_adapter.cc	Adapter class for the BLAS API	250
src/mtk_dense_matrix.cc		254
src/mtk_div_1d.cc	Implements the class Div1D	262
src/mtk_div_2d.cc	Implements the class Div2D	280
src/mtk_glpk_adapter.cc	Adapter class for the GLPK API	283
src/mtk_grad_1d.cc	Implements the class Grad1D	288
src/mtk_grad_2d.cc	Implements the class Grad2D	307
src/mtk_interp_1d.cc	Includes the implementation of the class Interp1D	310
src/mtk_lap_1d.cc	Includes the implementation of the class Lap1D	313
src/mtk_lap_2d.cc	Includes the implementation of the class Lap2D	318
src/mtk_lapack_adapter.cc	Adapter class for the LAPACK API	320
src/mtk_matrix.cc	Implementing the representation of a matrix in the MTK	329
src/mtk_tools.cc	Implements a execution tool manager class	333
src/mtk_uni_stg_grid_1d.cc	Implementation of an 1D uniform staggered grid	336
src/mtk_uni_stg_grid_2d.cc	Implementation of a 2D uniform staggered grid	340
tests/mtk_blas_adapter_test.cc	Test file for the mtk::BLASAdapter class	346
tests/mtk_dense_matrix_test.cc	Test file for the mtk::DenseMatrix class	349

tests/ mtk_div_1d_test.cc	
Testing the mimetic 1D divergence, constructed with the CBS algorithm	354
tests/ mtk_glpk_adapter_test.cc	
Test file for the mtk::GLPKAdapter class	358
tests/ mtk_grad_1d_test.cc	
Testing the mimetic 1D gradient, constructed with the CBS algorithm	360
tests/ mtk_grad_2d_test.cc	
Test file for the mtk::Grad2D class	365
tests/ mtk_interp_1d_test.cc	
Testing the 1D interpolation	367
tests/ mtk_lap_1d_test.cc	
Testing the 1D Laplacian operator	369
tests/ mtk_lapack_adapter_test.cc	
Test file for the mtk::LAPACKAdapter class	373
tests/ mtk_uni_stg_grid_1d_test.cc	
Test file for the mtk::UniStgGrid1D class	375
tests/ mtk_uni_stg_grid_2d_test.cc	
Test file for the mtk::UniStgGrid2D class	378

Chapter 14

Module Documentation

14.1 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::kDefaultTolerance` {1e-7f}
Considered tolerance for comparisons in numerical methods.
- const int `mtk::kDefaultOrderAccuracy` {2}
Default order of accuracy for mimetic operators.
- const float `mtk::kDefaultMimeticThreshold` {1e-6f}
Default tolerance for higher-order mimetic operators.
- const int `mtk::kCriticalOrderAccuracyDiv` {8}
At this order (and higher) we must use the CBSA to construct.
- const int `mtk::kCriticalOrderAccuracyGrad` {10}
At this order (and higher) we must use the CBSA to construct.

14.1.1 Detailed Description

Fundamental execution parameters and defined types.

14.1.2 Typedef Documentation

14.1.2.1 `mtk::Real`

Definition at line 83 of file [mtk_roots.h](#).

14.1.3 Variable Documentation

14.1.3.1 `mtk::kCriticalOrderAccuracyDiv {8}`

Definition at line 157 of file [mtk_roots.h](#).

14.1.3.2 `mtk::kCriticalOrderAccuracyGrad {10}`

Definition at line 166 of file [mtk_roots.h](#).

14.1.3.3 `mtk::kDefaultMimeticThreshold {1e-6f}`

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 147 of file [mtk_roots.h](#).

14.1.3.4 `mtk::kDefaultOrderAccuracy {2}`

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 133 of file [mtk_roots.h](#).

14.1.3.5 `mtk::kDefaultTolerance {1e-7f}`

Definition at line 121 of file [mtk_roots.h](#).

14.1.3.6 `mtk::kOne {1.0f}`

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 108 of file [mtk_roots.h](#).

14.1.3.7 `mtk::kZero {0.0f}`

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 107 of file [mtk_roots.h](#).

14.2 Enumerations.

Enumerations.

Enumerations

- enum `mtk::MatrixStorage` { `mtk::DENSE`, `mtk::BANDED`, `mtk::CRS` }
Considered matrix storage schemes to implement sparse matrices.
- enum `mtk::MatrixOrdering` { `mtk::ROW_MAJOR`, `mtk::COL_MAJOR` }
Considered matrix ordering (for Fortran purposes).
- enum `mtk::FieldNature` { `mtk::SCALAR`, `mtk::VECTOR` }
Nature of the field discretized in a given grid.
- enum `mtk::DirInterp` { `mtk::SCALAR_TO_VECTOR`, `mtk::VECTOR_TO_SCALAR` }
1D interpolation operator.

14.2.1 Detailed Description

Enumerations.

14.2.2 Enumeration Type Documentation

14.2.2.1 enum `mtk::DirInterp`

For now, it implements an arithmetic average.

Enumerator

`SCALAR_TO_VECTOR`

`VECTOR_TO_SCALAR`

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

14.2.2.2 enum `mtk::FieldNature`

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

See also

https://en.wikipedia.org/wiki/Scalar_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`.

14.2.2.3 enum mtk::MatrixOrdering

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

Enumerator

ROW_MAJOR Row-major ordering (C/C++).

COL_MAJOR Column-major ordering (Fortran).

Definition at line 95 of file [mtk_enums.h](#).

14.2.2.4 enum mtk::MatrixStorage

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](#).

14.3 Execution tools.

Tools to ensure execution correctness.

Classes

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

14.3.1 Detailed Description

Tools to ensure execution correctness.

14.4 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.

14.4.1 Detailed Description

Fundamental data structures.

14.5 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.

14.5.1 Detailed Description

Adapter classes and auxiliary numerical methods.

14.6 Grids.

Uniform rectangular staggered grids.

Classes

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

14.6.1 Detailed Description

Uniform rectangular staggered grids.

14.7 Mimetic operators.

Mimetic operators.

Classes

- class [mtk::Div1D](#)
Implements a 1D mimetic divergence operator.
- class [mtk::Grad1D](#)
Implements a 1D mimetic gradient operator.
- class [mtk::Interp1D](#)
Implements a 1D interpolation operator.
- class [mtk::Lap1D](#)
Implements a 1D mimetic Laplacian operator.
- class [mtk::Quad1D](#)
Implements a 1D mimetic quadrature.

14.7.1 Detailed Description

Mimetic operators.

Chapter 15

Namespace Documentation

15.1 mtk Namespace Reference

Mimetic Methods Toolkit namespace.

Classes

- class [BCDesc1D](#)
- class [BCDesc2D](#)
- class [BLASAdapter](#)
Adapter class for the BLAS API.
- class [DenseMatrix](#)
Defines a common dense matrix, using a 1D array.
- class [Div1D](#)
Implements a 1D mimetic divergence operator.
- class [Div2D](#)
- class [GLPKAdapter](#)
Adapter class for the GLPK API.
- class [Grad1D](#)
Implements a 1D mimetic gradient operator.
- class [Grad2D](#)
- class [Interp1D](#)
Implements a 1D interpolation operator.
- class [Interp2D](#)
- class [Lap1D](#)
Implements a 1D mimetic Laplacian operator.
- class [Lap2D](#)
- 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 [Tools](#)

Tool manager class.

- class [UniStgGrid1D](#)

Uniform 1D Staggered Grid.

- class [UniStgGrid2D](#)

Uniform 2D Staggered Grid.

Typedefs

- typedef float [Real](#)

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

Enumerations

- enum [MatrixStorage](#) { [DENSE](#), [BANDED](#), [CRS](#) }

Considered matrix storage schemes to implement sparse matrices.

- enum [MatrixOrdering](#) { [ROW_MAJOR](#), [COL_MAJOR](#) }

Considered matrix ordering (for Fortran purposes).

- enum [FieldNature](#) { [SCALAR](#), [VECTOR](#) }

Nature of the field discretized in a given grid.

- enum [DirInterp](#) { [SCALAR_TO_VECTOR](#), [VECTOR_TO_SCALAR](#) }

1D 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)

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 [kDefaultTolerance](#) {1e-7f}
Considered tolerance for comparisons in numerical methods.
- const int [kDefaultOrderAccuracy](#) {2}
Default order of accuracy for mimetic operators.
- const float [kDefaultMimeticThreshold](#) {1e-6f}
Default tolerance for higher-order mimetic operators.
- const int [kCriticalOrderAccuracyDiv](#) {8}
At this order (and higher) we must use the CBSA to construct.
- const int [kCriticalOrderAccuracyGrad](#) {10}
At this order (and higher) we must use the CBSA to construct.

15.1.1 Function Documentation

15.1.1.1 `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](#).

15.1.1.2 `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](#).

15.1.1.3 `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](#).

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

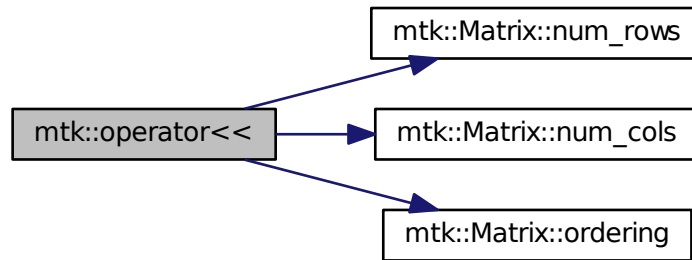
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

Definition at line 73 of file [mtk_lap_1d.cc](#).

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

Definition at line 77 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



15.1.1.6 `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 79 of file [mtk_grad_1d.cc](#).

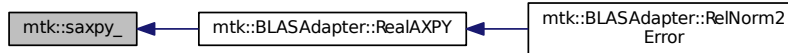
15.1.1.7 `std::ostream& mtk::operator<< (std::ostream & stream, mtk::Div1D & in)`

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 79 of file [mtk_div_1d.cc](#).

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

Here is the caller graph for this function:



15.1.1.9 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

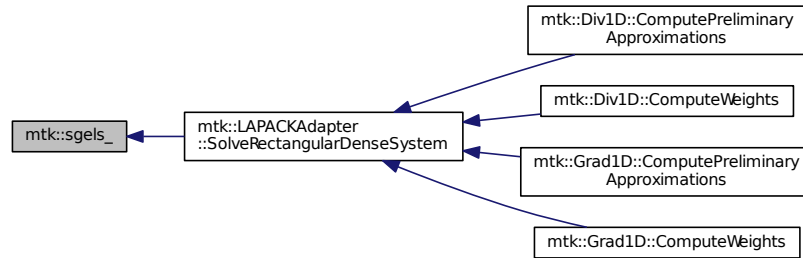
<http://www.math.utah.edu/software/lapack/lapack-s/sgels.html>

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. $lda \geq \max(1, m)$.

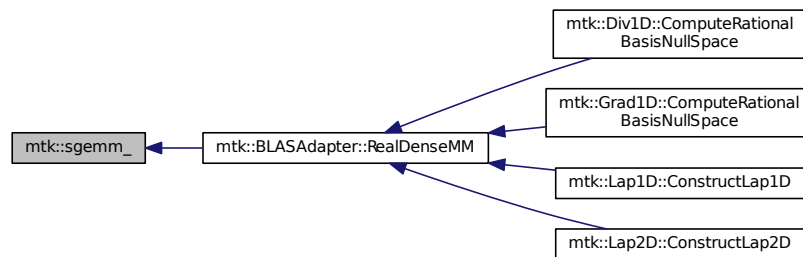
in, out	<i>b</i>	On entry, matrix b of right-hand side vectors.
in	<i>ldb</i>	The leading dimension of b. $ldb \geq \max(1, m, n)$.
in, out	<i>work</i>	On exit, if <i>info</i> = 0, <i>work</i> (1) is optimal <i>lwork</i> .
in, out	<i>lwork</i>	The dimension of the array <i>work</i> .
in, out	<i>info</i>	If <i>info</i> = 0, then successful exit.

Here is the caller graph for this function:



15.1.1.10 `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:



15.1.1.11 `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:



15.1.1.12 `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 C * Q` `TRANS = 'T': Q**T * C 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

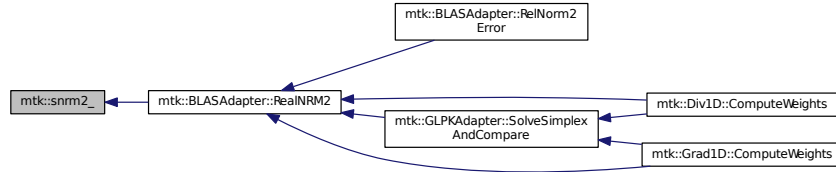
Parameters

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

15.1.1.13 `void mtk::sgesv_(int * n, int * nrhs, Real * a, int * lda, int * ipiv, Real * b, int * ldb, int * info)`

15.1.1.14 float mtk::snrm2_ (int * n, float * x, int * incx)

Here is the caller graph for this function:



15.1.1.15 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': $Q * C * Q$ 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

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 >= 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.

<code>in, out</code>	<i>info</i>	info = 0: successful exit.
----------------------	-------------	----------------------------

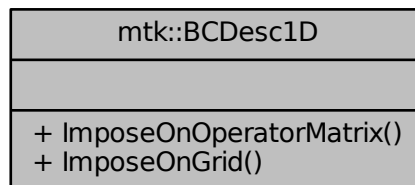
Chapter 16

Class Documentation

16.1 mtk::BCDesc1D Class Reference

```
#include <mtk_bc_desc_1d.h>
```

Collaboration diagram for mtk::BCDesc1D:



Static Public Member Functions

- static void `ImposeOnOperatorMatrix` (`DenseMatrix` &matrix, const std::vector< `Real` > &west, const std::vector< `Real` > &east)
Enforces the condition on the operator represented as matrix.
- static void `ImposeOnGrid` (`UniStgGrid1D` &grid, const `Real` &omega, const `Real` &epsilon)
Enforces the condition on the grid.

16.1.1 Detailed Description

Definition at line 68 of file `mtk_bc_desc_1d.h`.

16.1.2 Member Function Documentation

16.1.2.1 `void mtk::BCDesc1D::ImposeOnGrid (mtk::UniStgGrid1D & grid, const Real & omega, const Real & epsilon)`
[static]

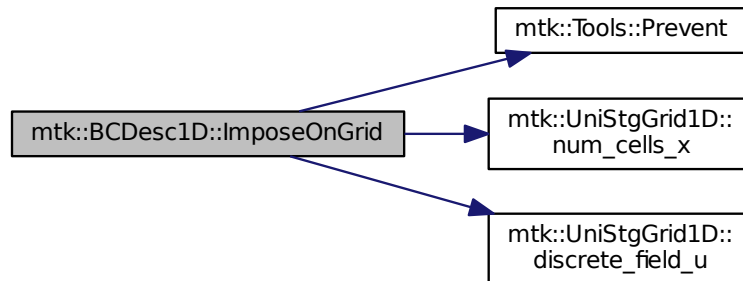
Parameters

in, out	<i>grid</i>	Input grid.
in	<i>west</i>	Array of values for the west boundary.
in	<i>east</i>	Array of values for the east boundary.

1. Assign the west condition.
2. Assign the east condition.

Definition at line 88 of file [mtk_bc_desc_1d.cc](#).

Here is the call graph for this function:



16.1.2.2 `void mtk::BCDesc1D::ImposeOnOperatorMatrix (mtk::DenseMatrix & matrix, const std::vector< Real > & west, const std::vector< Real > & east) [static]`

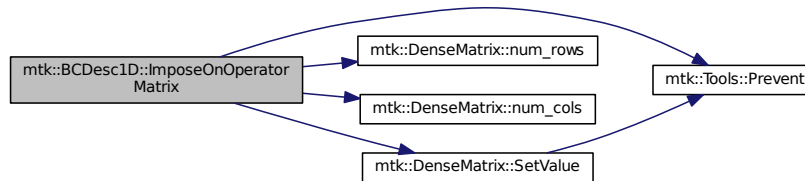
Parameters

in, out	<i>matrix</i>	Input operator.
in	<i>west</i>	Array of values for the west boundary.
in	<i>east</i>	Array of values for the east boundary.

1. Assign the west array.
2. Assign the east array.

Definition at line 61 of file [mtk_bc_desc_1d.cc](#).

Here is the call graph for this function:



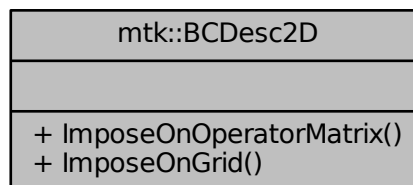
The documentation for this class was generated from the following files:

- [include/mtk_bc_desc_1d.h](#)
- [src/mtk_bc_desc_1d.cc](#)

16.2 mtk::BCDesc2D Class Reference

```
#include <mtk_bc_desc_2d.h>
```

Collaboration diagram for mtk::BCDesc2D:



Static Public Member Functions

- static void [ImposeOnOperatorMatrix](#) (const [DenseMatrix](#) &matrix, [Real](#)(*west)(int ii, int jj), [Real](#)(*east)(int ii, int jj), [Real](#)(*south)(int ii, int jj), [Real](#)(*north)(int ii, int jj))
Enforces the condition on the operator represented as matrix.
- static void [ImposeOnGrid](#) (const [UniStgGrid2D](#) &grid, [Real](#)(*west)([Real](#) xx, [Real](#) yy), [Real](#)(*east)([Real](#) xx, [Real](#) yy), [Real](#)(*south)([Real](#) xx, [Real](#) yy), [Real](#)(*north)([Real](#) xx, [Real](#) yy))
Enforces the condition on the grid.

16.2.1 Detailed Description

Definition at line 66 of file [mtk_bc_desc_2d.h](#).

16.2.2 Member Function Documentation

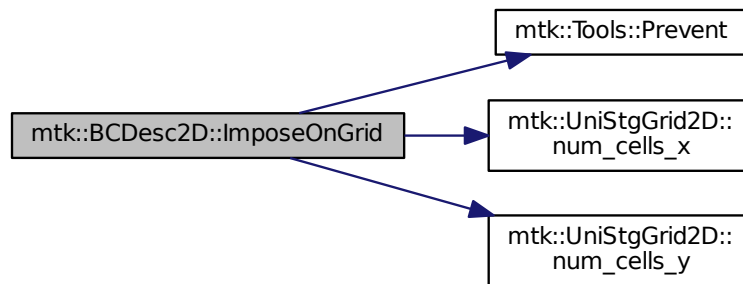
16.2.2.1 `void mtk::BCDesc2D::ImposeOnGrid (const UniStgGrid2D & grid, Real(*)(Real xx, Real yy) west, Real(*)(Real xx, Real yy) east, Real(*)(Real xx, Real yy) south, Real(*)(Real xx, Real yy) north) [static]`

Parameters

<i>in, out</i>	<i>matrix</i>	Input operator.
<i>in</i>	<i>west</i>	Pointer to function returning west coefficient at (xx,yy).
<i>in</i>	<i>east</i>	Pointer to function returning east coefficient at (xx,yy).
<i>in</i>	<i>south</i>	Pointer to function returning south coefficient at (xx,yy).
<i>in</i>	<i>north</i>	Pointer to function returning north coefficient at (xx,yy).

Definition at line 78 of file [mtk_bc_desc_2d.cc](#).

Here is the call graph for this function:



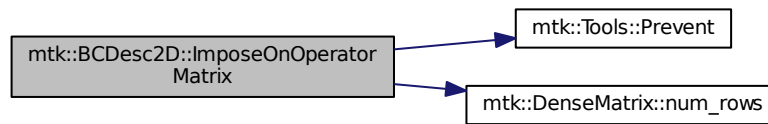
16.2.2.2 `void mtk::BCDesc2D::ImposeOnOperatorMatrix (const DenseMatrix & matrix, mtk::Real(*)(int ii, int jj) west, mtk::Real(*)(int ii, int jj) east, mtk::Real(*)(int ii, int jj) south, mtk::Real(*)(int ii, int jj) north) [static]`

Parameters

<i>in, out</i>	<i>matrix</i>	Input operator.
<i>in</i>	<i>west</i>	Pointer to function returning west coefficient at (ii,jj).
<i>in</i>	<i>east</i>	Pointer to function returning east coefficient at (ii,jj).
<i>in</i>	<i>south</i>	Pointer to function returning south coefficient at (ii,jj).
<i>in</i>	<i>north</i>	Pointer to function returning north coefficient at (ii,jj).

Definition at line 61 of file [mtk_bc_desc_2d.cc](#).

Here is the call graph for this function:



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

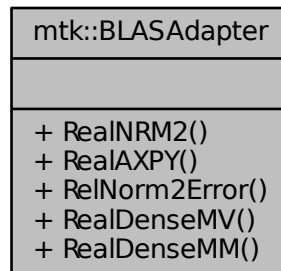
- [include/mtk_bc_desc_2d.h](#)
- [src/mtk_bc_desc_2d.cc](#)

16.3 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](#) [RealINRM2](#) ([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.

16.3.1 Detailed Description

This class contains a collection of static classes, 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/>

Definition at line 96 of file [mtk_blas_adapter.h](#).

16.3.2 Member Function Documentation

16.3.2.1 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

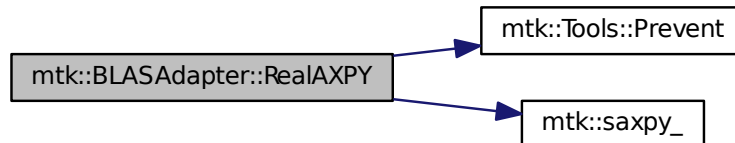
in	<i>alpha</i>	Scalar of the first array.
in	<i>xx</i>	First array.
in	<i>yy</i>	Second array.
in	<i>in_length</i>	Lengths of the given arrays.

Returns

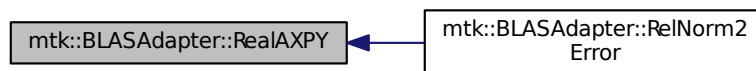
Norm-2 of the given array.

Definition at line 339 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.3.2.2 `mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM (mtk::DenseMatrix & aa, mtk::DenseMatrix & bb)`
`[static]`

Performs:

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

Parameters

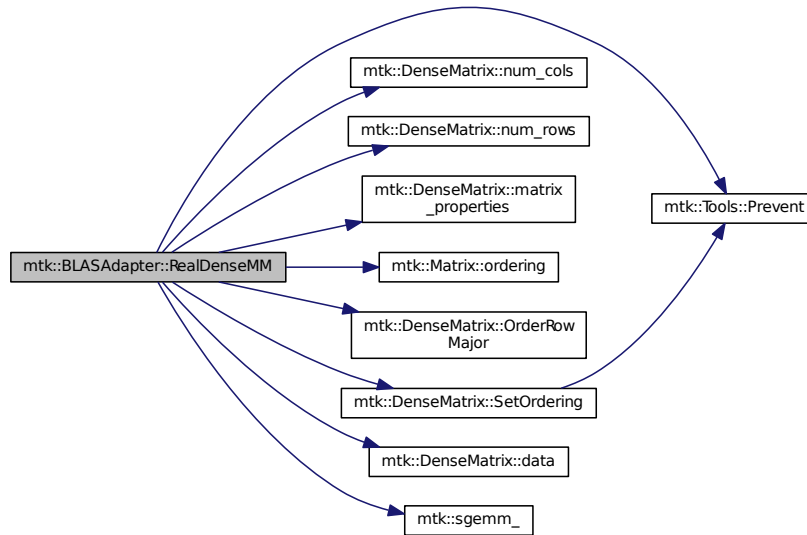
in	<i>aa</i>	First matrix.
in	<i>bb</i>	Second matrix.

See also

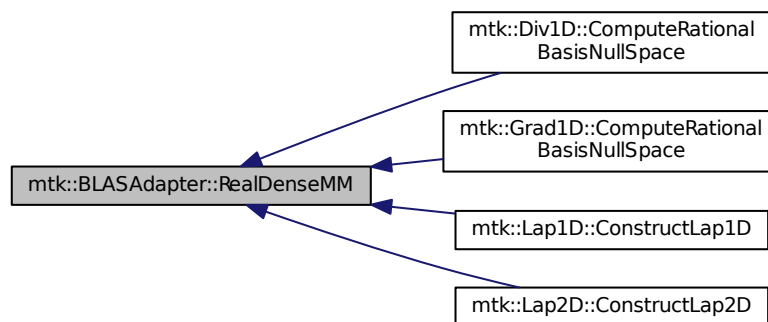
<http://ejspeiro.github.io/Netlib-and-CPP/>

Definition at line 409 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.3.2.3 `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

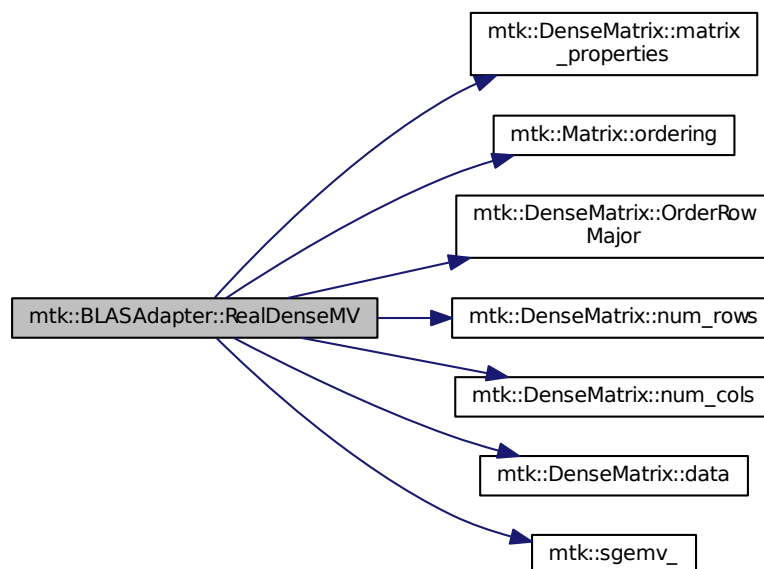
in	<i>alpha</i>	First scalar.
in	<i>aa</i>	Given matrix.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).

See also

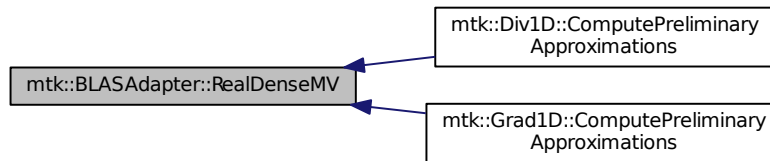
<http://ejspeiro.github.io/Netlib-and-CPP/>

Definition at line 378 of file `mtk_blas_adapter.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

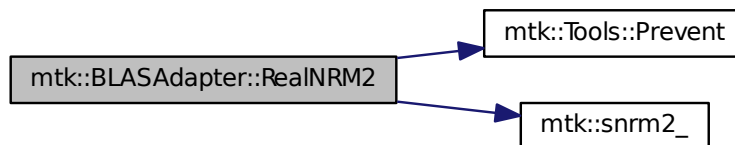
<code>in</code>	<code>in</code>	Input array.
<code>in</code>	<code>in_length</code>	Length of the array.

Returns

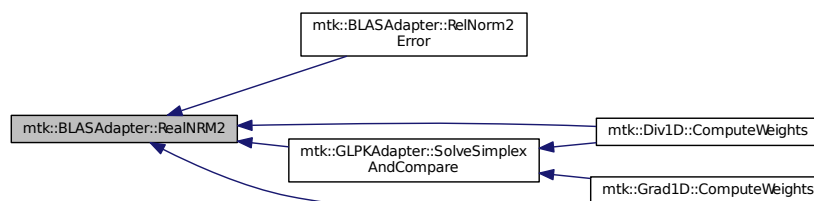
Norm-2 of the given array.

Definition at line 324 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.3.2.5 `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

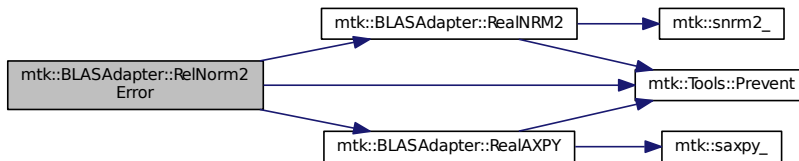
in	<i>known</i>	Array containing the computed solution.
in	<i>computed</i>	Array containing the known solution (ref. solution).

Returns

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

Definition at line 358 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:

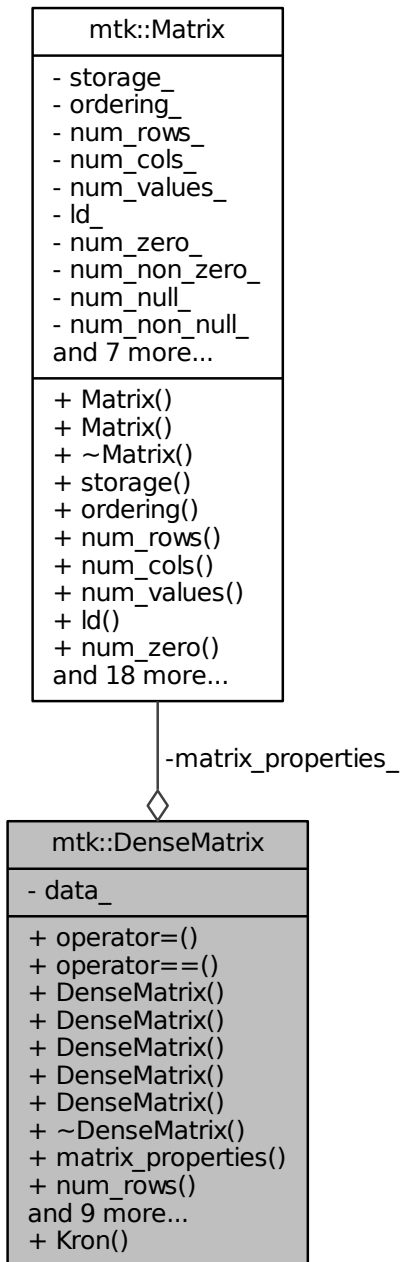
- [include/mtk_blas_adapter.h](#)
- [src/mtk_blas_adapter.cc](#)

16.4 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 *gen, const int &gen_length, const int &pro_length, const bool &transpose)`

Construct a dense Vandermonde matrix.

- `~DenseMatrix ()`

Destructor.

- `Matrix matrix_properties () const`

Provides access to the matrix data.

- `int num_rows () const`

Gets the number of rows.

- `int num_cols () const`

Gets the number of columns.

- `Real * data () const`

Provides access to the matrix value array.

- `void SetOrdering (mtk::MatrixOrdering oo)`

Sets the ordering of the matrix.

- `Real GetValue (const int &row_coord, const int &col_coord) const`

Gets a value on the given coordinates.

- `void SetValue (const int &row_coord, const int &col_coord, const Real &val)`

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 (std::string filename)`

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).

16.4.1 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.

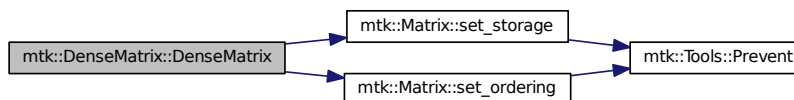
Definition at line 92 of file [mtk_dense_matrix.h](#).

16.4.2 Constructor & Destructor Documentation

16.4.2.1 `mtk::DenseMatrix::DenseMatrix ()`

Definition at line 162 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



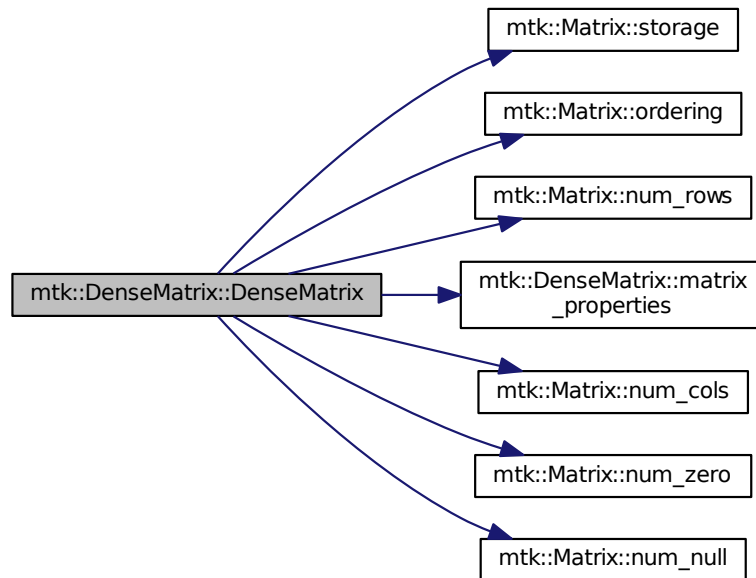
16.4.2.2 `mtk::DenseMatrix::DenseMatrix (const DenseMatrix &in)`

Parameters

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

Definition at line 168 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



16.4.2.3 `mtk::DenseMatrix::DenseMatrix (const int & num_rows, const int & num_cols)`

Parameters

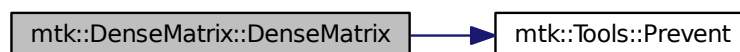
in	<i>num_rows</i>	Number of rows of the required matrix.
in	<i>num_cols</i>	Number of rows of the required matrix.

Exceptions

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

Definition at line 201 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



16.4.2.4 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 223 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



16.4.2.5 mtk::DenseMatrix::DenseMatrix (const Real * *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

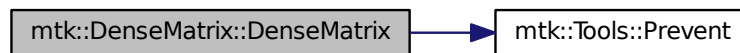
in	<i>gen</i>	Given generator vector.
in	<i>gen_length</i>	Length generator vector.
in	<i>pro_length</i>	Length the progression.
in	<i>transpose</i>	Should the transpose be created instead?

Exceptions

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

Definition at line [264](#) of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



16.4.2.6 mtk::DenseMatrix::~~DenseMatrix ()

Definition at line [312](#) of file [mtk_dense_matrix.cc](#).

16.4.3 Member Function Documentation

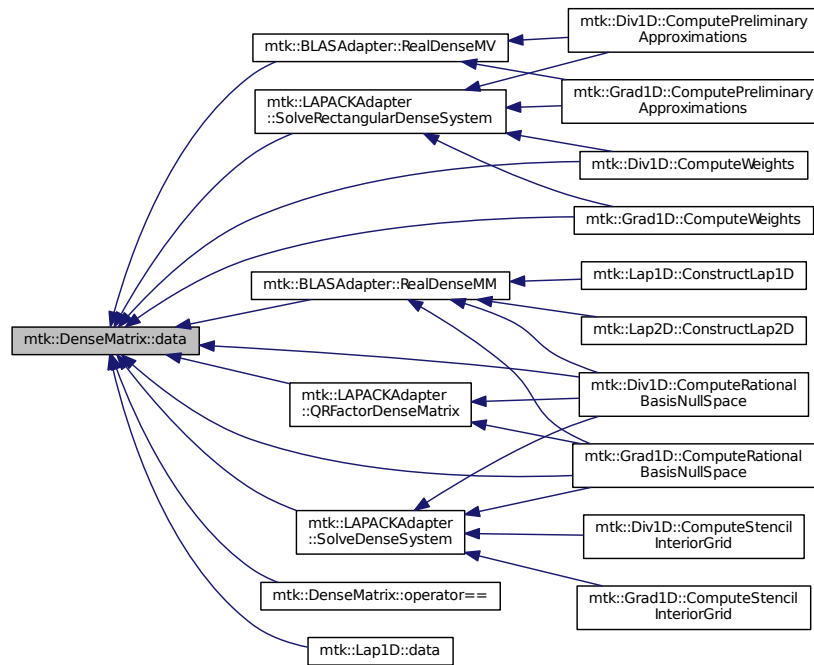
16.4.3.1 mtk::Real * mtk::DenseMatrix::data () const

Returns

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

Definition at line 343 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



16.4.3.2 mtk::Real mtk::DenseMatrix::GetValue (const int & row_coord, const int & col_coord) const

Parameters

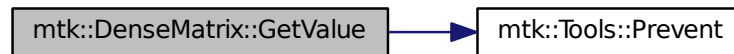
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.

Returns

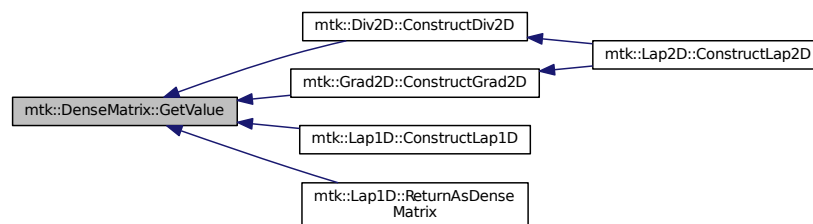
The required value at the specified coordinates.

Definition at line 348 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.4.3.3 `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.

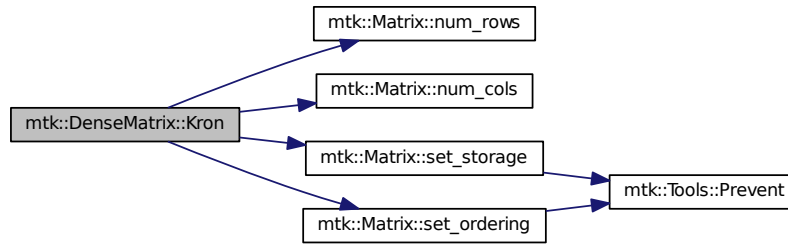
Exceptions

<code>std::bad_alloc</code>

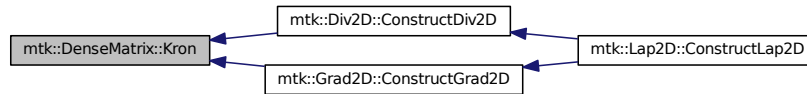
Todo Implement Kronecker product using the BLAS.

Definition at line 490 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



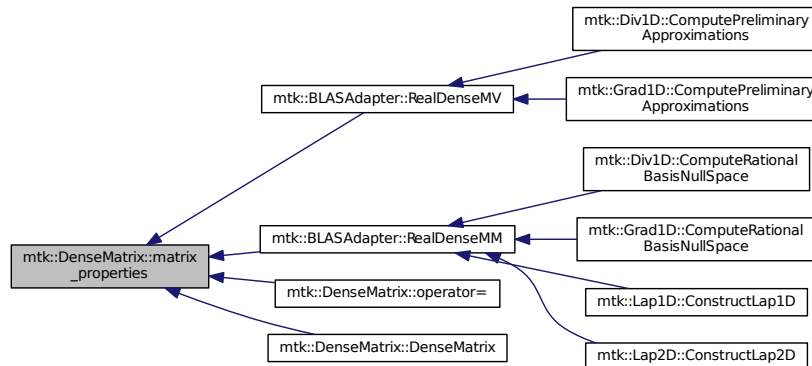
16.4.3.4 mtk::Matrix mtk::DenseMatrix::matrix_properties () const

Returns

Pointer to a [Matrix](#).

Definition at line 318 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



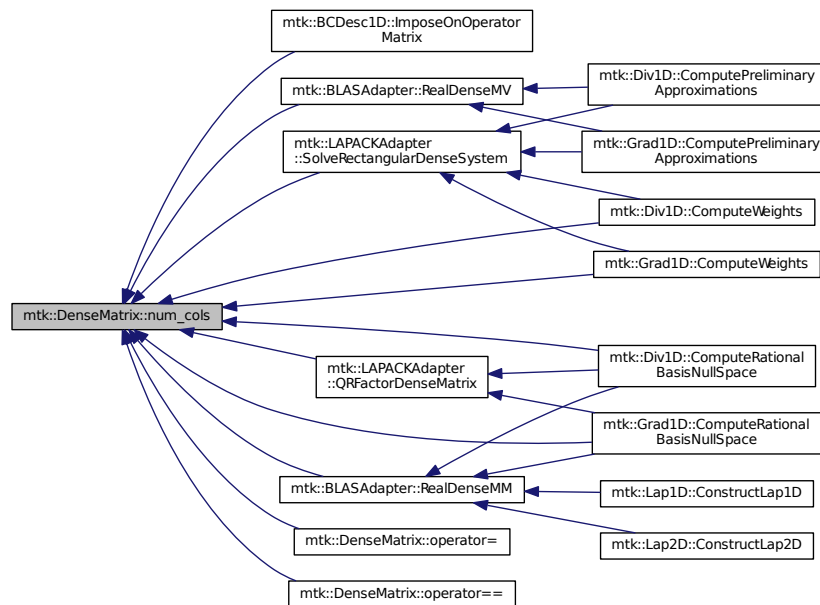
16.4.3.5 `int mtk::DenseMatrix::num_cols () const`

Returns

Number of columns of the matrix.

Definition at line 338 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



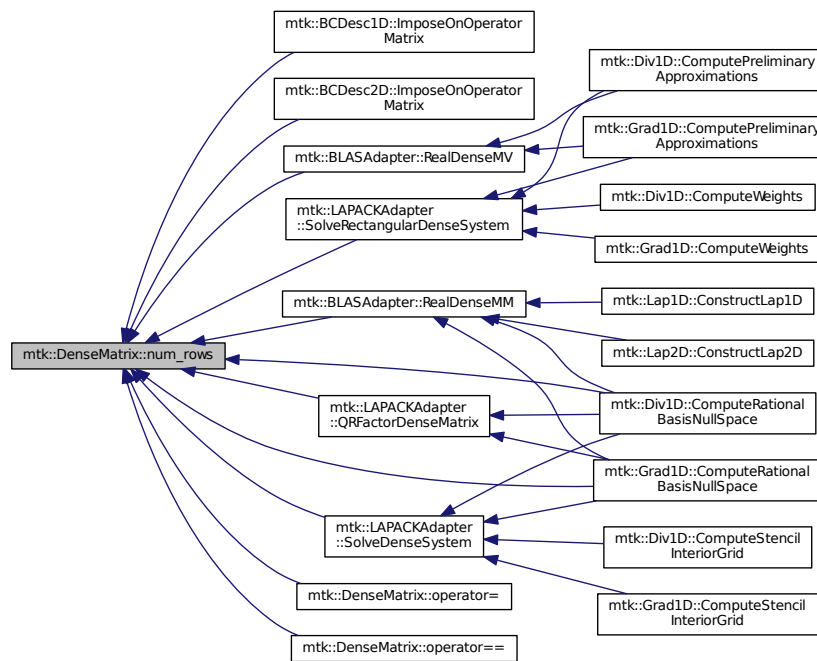
16.4.3.6 `int mtk::DenseMatrix::num_rows () const`

Returns

Number of rows of the matrix.

Definition at line 333 of file [mtk_dense_matrix.cc](#).

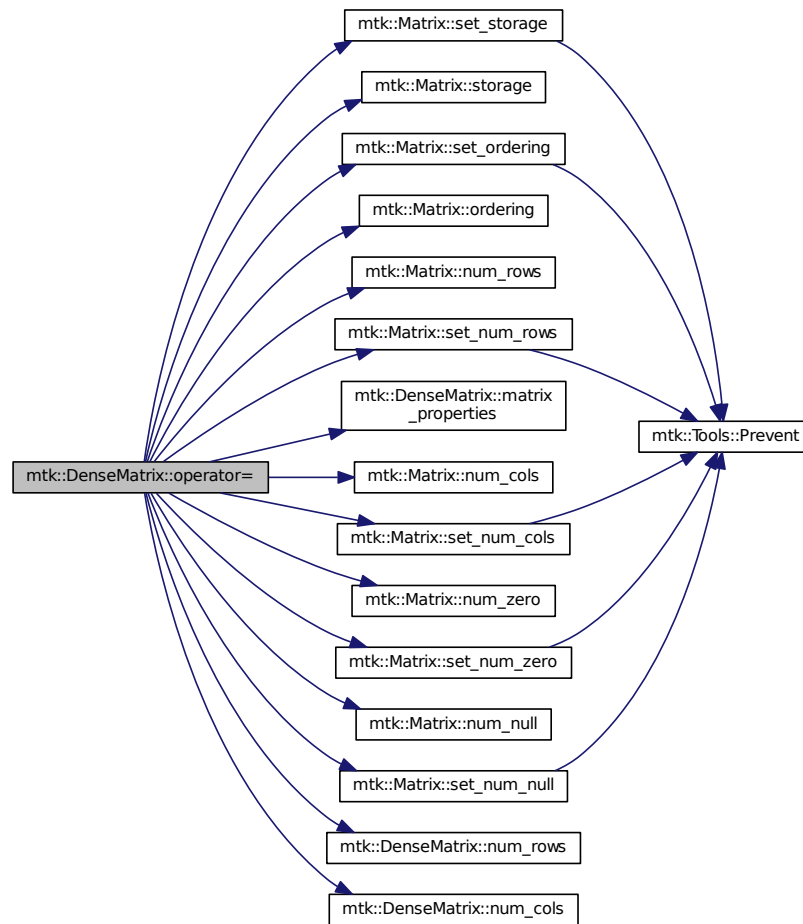
Here is the caller graph for this function:



16.4.3.7 mtk::DenseMatrix & mtk::DenseMatrix::operator= (const DenseMatrix & in)

Definition at line 100 of file [mtk_dense_matrix.cc](#).

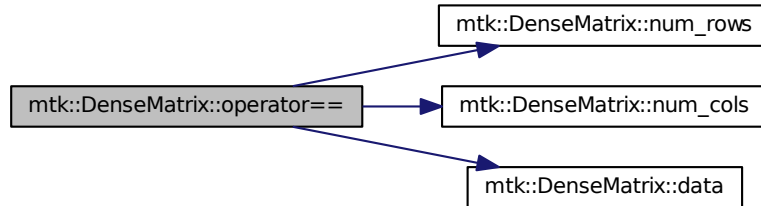
Here is the call graph for this function:



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

Definition at line 141 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:

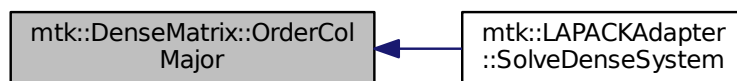


16.4.3.9 `void mtk::DenseMatrix::OrderColMajor ()`

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

Definition at line 451 of file `mtk_dense_matrix.cc`.

Here is the caller graph for this function:

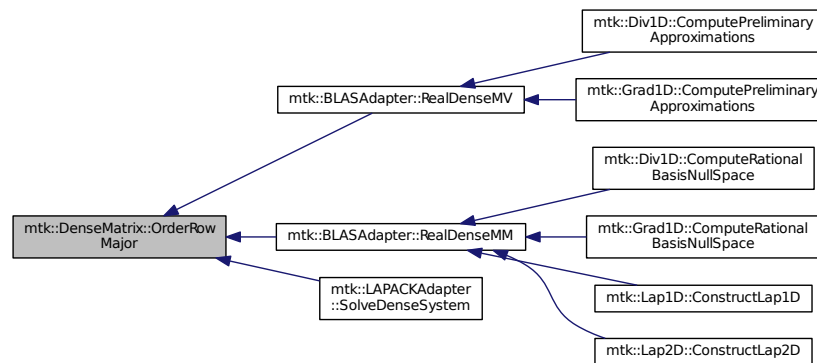


16.4.3.10 `void mtk::DenseMatrix::OrderRowMajor ()`

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

Definition at line 410 of file `mtk_dense_matrix.cc`.

Here is the caller graph for this function:



16.4.3.11 `void mtk::DenseMatrix::SetOrdering (mtk::MatrixOrdering oo)`

Parameters

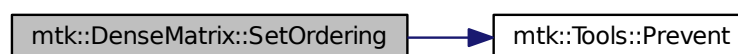
in	oo	Ordering.
----	----	-----------

Returns

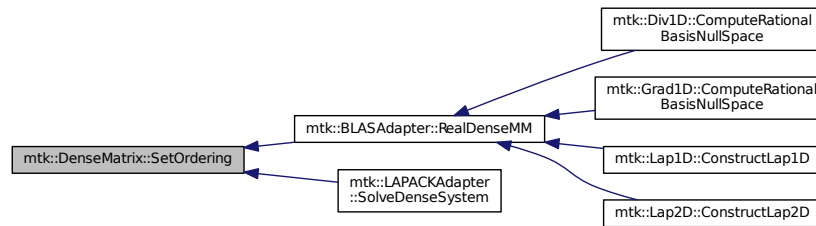
The required value at the specified coordinates.

Definition at line 323 of file `mtk_dense_matrix.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



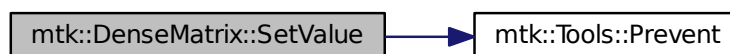
16.4.3.12 `void mtk::DenseMatrix::SetValue (const int & row_coord, const int & col_coord, const Real & val)`

Parameters

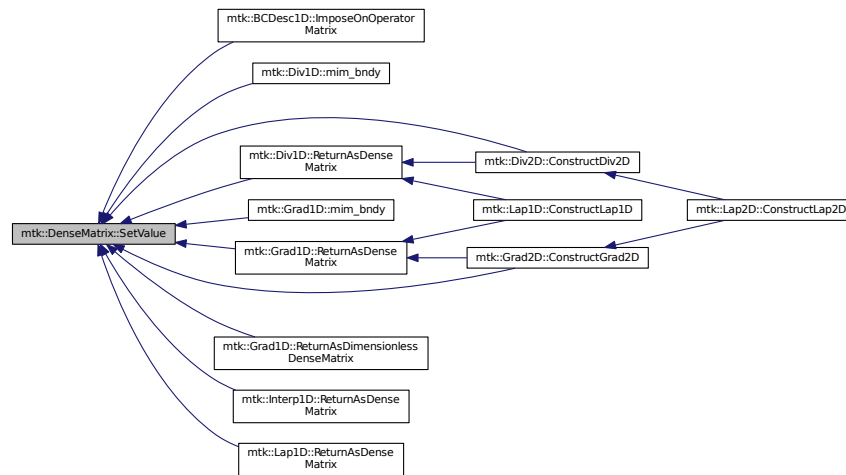
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.
in	<i>val</i>	Row Actual value to be inserted.

Definition at line 360 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:

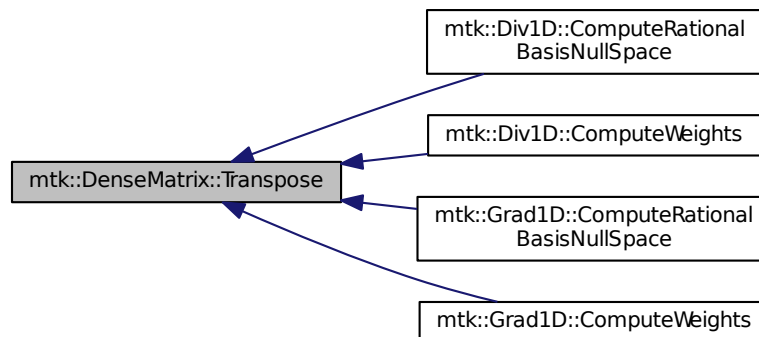


16.4.3.13 void mtk::DenseMatrix::Transpose ()

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

Definition at line 373 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



16.4.3.14 bool mtk::DenseMatrix::WriteToFile (std::string filename)

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 531 of file [mtk_dense_matrix.cc](#).

16.4.4 Friends And Related Function Documentation

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

Definition at line 77 of file [mtk_dense_matrix.cc](#).

16.4.5 Member Data Documentation

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

Definition at line 284 of file [mtk_dense_matrix.h](#).

16.4.5.2 `Matrix mtk::DenseMatrix::matrix_properties_` `[private]`

Definition at line 282 of file [mtk_dense_matrix.h](#).

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

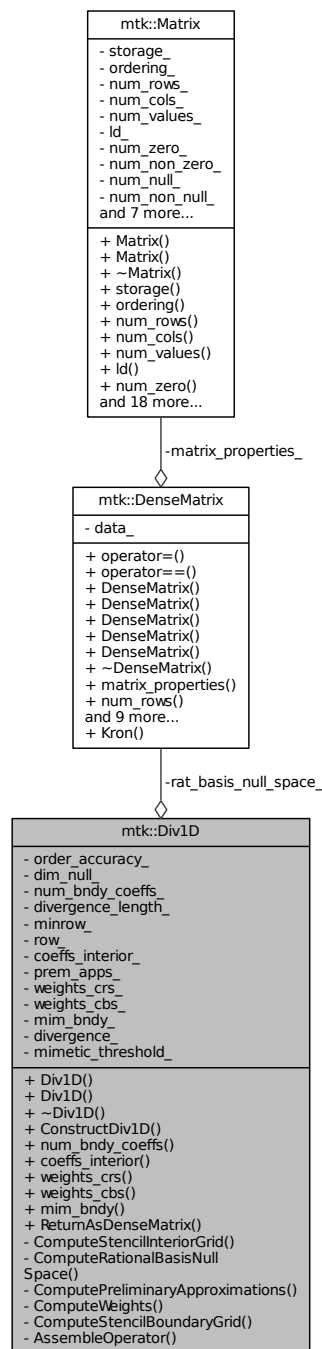
- [include/mtk_dense_matrix.h](#)
- [src/mtk_dense_matrix.cc](#)

16.5 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_coefs](#) () const
- Returns how many coefficients are approximating at the boundary.*
- [Real](#) * [coefs_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)
- Return the operator as a 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_coefs_](#)
- Req. coefs. 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.
- `Real mimetic_threshold_`
< Mimetic threshold.

Friends

- `std::ostream & operator<< (std::ostream &stream, Div1D &in)`
Output stream operator for printing.

16.5.1 Detailed Description

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

Definition at line 81 of file `mtk_div_1d.h`.

16.5.2 Constructor & Destructor Documentation

16.5.2.1 `mtk::Div1D::Div1D ()`

Definition at line 125 of file `mtk_div_1d.cc`.

16.5.2.2 `mtk::Div1D::Div1D (const Div1D &div)`

Parameters

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

Definition at line 140 of file `mtk_div_1d.cc`.

16.5.2.3 `mtk::Div1D::~~Div1D ()`

Definition at line 155 of file `mtk_div_1d.cc`.

16.5.3 Member Function Documentation

16.5.3.1 `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 1333 of file [mtk_div_1d.cc](#).

16.5.3.2 `mtk::Real * mtk::Div1D::coeffs_interior () const`

Returns

Coefficients for the interior of the grid.

Definition at line 320 of file [mtk_div_1d.cc](#).

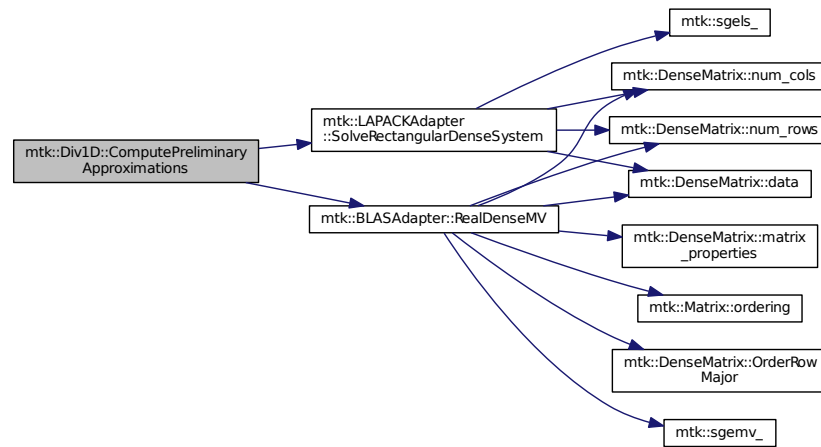
16.5.3.3 `bool mtk::Div1D::ComputePreliminaryApproximations (void) [private]`

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 688 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



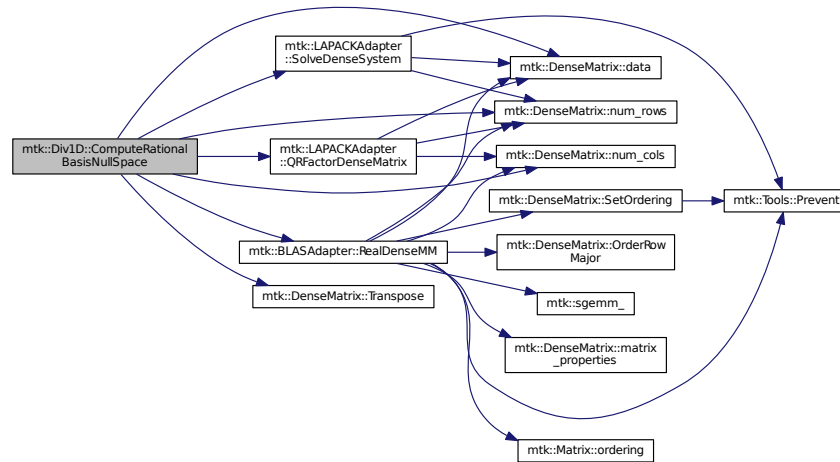
16.5.3.4 `bool mtk::Div1D::ComputeRationalBasisNullSpace (void) [private]`

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 512 of file `mtk_div_1d.cc`.

Here is the call graph for this function:



16.5.3.5 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 1234 of file [mtk_div_1d.cc](#).

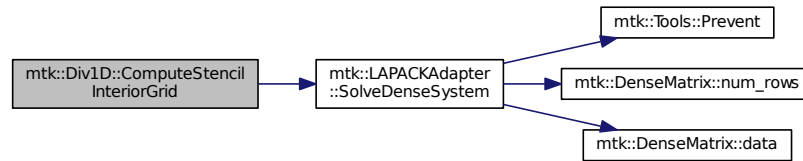
16.5.3.6 bool mtk::Div1D::ComputeStencilInteriorGrid (void) [private]

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 413 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



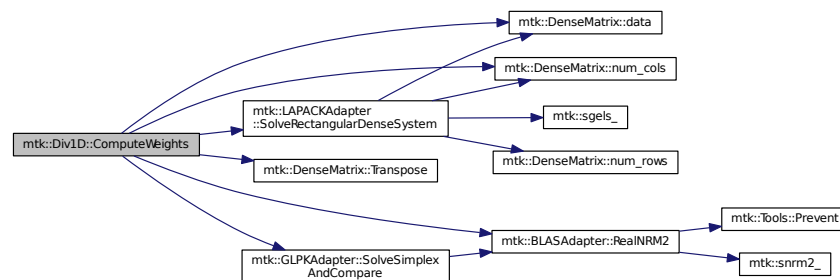
16.5.3.7 bool mtk::Div1D::ComputeWeights (void) [private]

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{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 Φ matrix.
8. Apply solution found from brute force search.

Definition at line 908 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



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

Returns

Success of the construction.

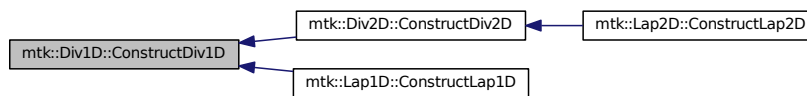
1. Compute stencil for the interior cells.
2. Compute a rational basis for the null-space for the first matrix.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 176 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



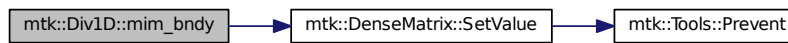
16.5.3.9 `mtk::DenseMatrix mtk::Div1D::mim_bndy () const`

Returns

Collection of mimetic approximations at the boundary.

Definition at line 336 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



16.5.3.10 `int mtk::Div1D::num_bndy_coeffs () const`

Returns

How many coefficients are approximating at the boundary.

Definition at line 315 of file [mtk_div_1d.cc](#).

16.5.3.11 `mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid)`

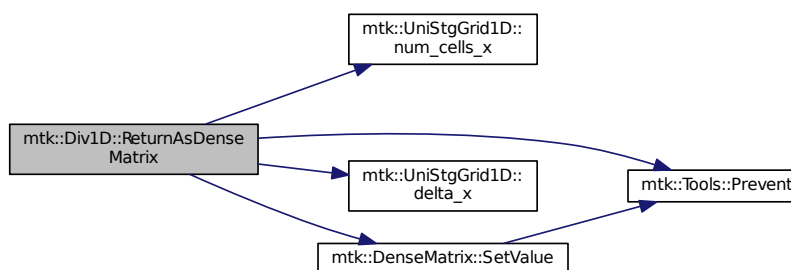
Returns

The operator as a dense matrix.

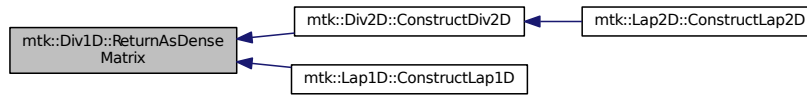
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 351 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.5.3.12 mtk::Real * mtk::Div1D::weights_cbs (void) const

Returns

Collection of weights as computed by the CBSA.

Definition at line 330 of file [mtk_div_1d.cc](#).

16.5.3.13 mtk::Real * mtk::Div1D::weights_crs (void) const

Returns

Collection of weights as computed by the CRSA.

Definition at line 325 of file [mtk_div_1d.cc](#).

16.5.4 Friends And Related Function Documentation

16.5.4.1 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 79 of file [mtk_div_1d.cc](#).

16.5.5 Member Data Documentation

16.5.5.1 Real* mtk::Div1D::coeffs_interior_ [private]

Definition at line 202 of file [mtk_div_1d.h](#).

16.5.5.2 int mtk::Div1D::dim_null_ [private]

Definition at line 194 of file [mtk_div_1d.h](#).

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

Definition at line 207 of file [mtk_div_1d.h](#).

16.5.5.4 `int mtk::Div1D::divergence_length_` [private]

Definition at line 196 of file [mtk_div_1d.h](#).

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

Definition at line 206 of file [mtk_div_1d.h](#).

16.5.5.6 `Real mtk::Div1D::mimetic_threshold_` [private]

Definition at line 209 of file [mtk_div_1d.h](#).

16.5.5.7 `int mtk::Div1D::minrow_` [private]

Definition at line 197 of file [mtk_div_1d.h](#).

16.5.5.8 `int mtk::Div1D::num_bndy_coeffs_` [private]

Definition at line 195 of file [mtk_div_1d.h](#).

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

Definition at line 193 of file [mtk_div_1d.h](#).

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

Definition at line 203 of file [mtk_div_1d.h](#).

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

Definition at line 200 of file [mtk_div_1d.h](#).

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

Definition at line 198 of file [mtk_div_1d.h](#).

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

Definition at line 205 of file [mtk_div_1d.h](#).

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

Definition at line 204 of file [mtk_div_1d.h](#).

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

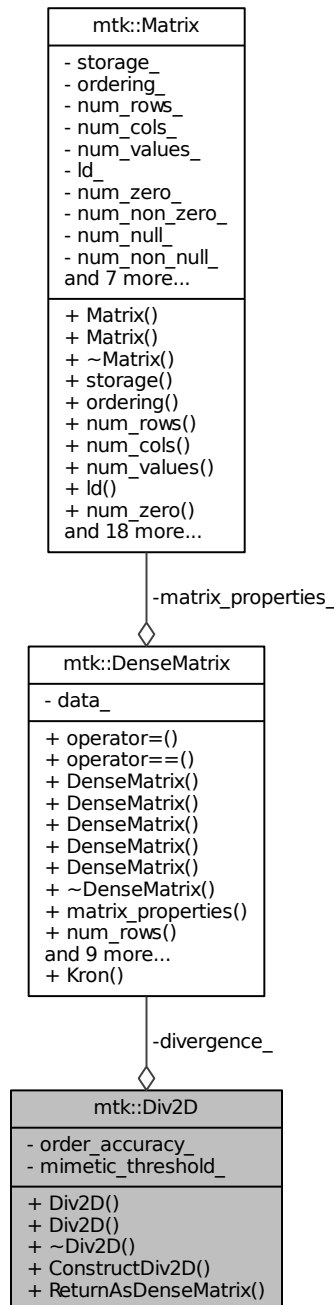
- [include/mtk_div_1d.h](#)

- [src/mtk_div_1d.cc](#)

16.6 mtk::Div2D Class Reference

```
#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](#) ()

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) divergence_

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

- [Real](#) [mimetic_threshold_](#)

Mimetic Threshold.

16.6.1 Detailed Description

Definition at line 66 of file [mtk_div_2d.h](#).

16.6.2 Constructor & Destructor Documentation

16.6.2.1 mtk::Div2D::Div2D ()

Definition at line 69 of file [mtk_div_2d.cc](#).

16.6.2.2 mtk::Div2D::Div2D (const Div2D &div)

Parameters

in	div	Given divergence.
--------------------	---------------------	-------------------

Definition at line 73 of file [mtk_div_2d.cc](#).

16.6.2.3 mtk::Div2D::~~Div2D ()

Definition at line 77 of file [mtk_div_2d.cc](#).

16.6.3 Member Function Documentation

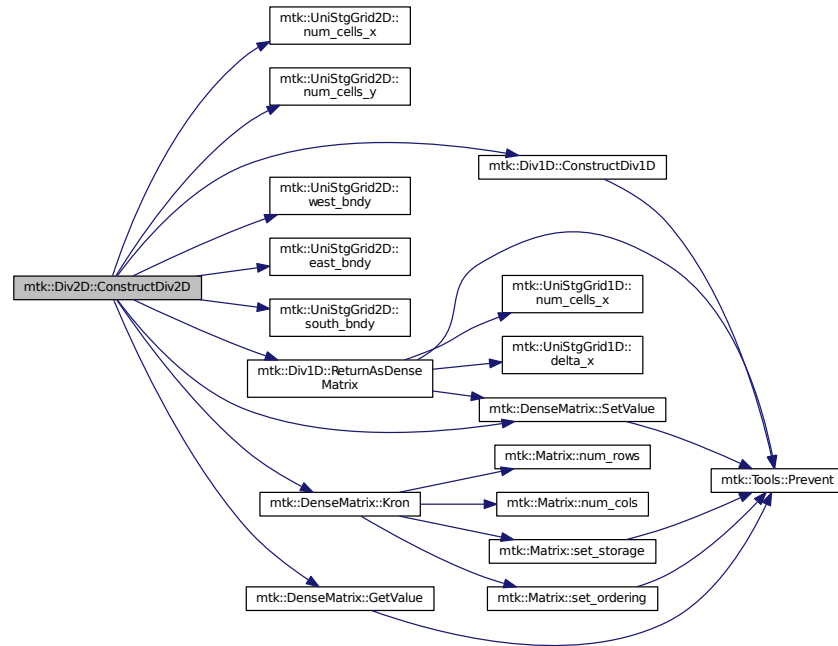
16.6.3.1 bool mtk::Div2D::ConstructDiv2D (const UniStgGrid2D &grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)

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:



16.6.3.2 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix ()

Returns

The operator as a dense matrix.

Definition at line 145 of file [mtk_div_2d.cc](#).

Here is the caller graph for this function:

**16.6.4 Member Data Documentation****16.6.4.1 DenseMatrix mtk::Div2D::divergence_ [private]**

Definition at line 98 of file [mtk_div_2d.h](#).

16.6.4.2 Real mtk::Div2D::mimetic_threshold_ [private]

Definition at line 102 of file [mtk_div_2d.h](#).

16.6.4.3 int mtk::Div2D::order_accuracy_ [private]

Definition at line 100 of file [mtk_div_2d.h](#).

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

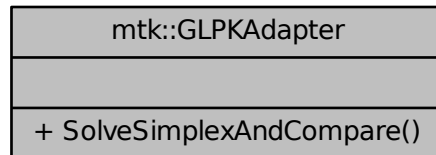
- [include/mtk_div_2d.h](#)
- [src/mtk_div_2d.cc](#)

16.7 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.

16.7.1 Detailed Description

This class contains a collection of static classes, 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 101 of file `mtk_glpk_adapter.h`.

16.7.2 Member Function Documentation

16.7.2.1 `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.

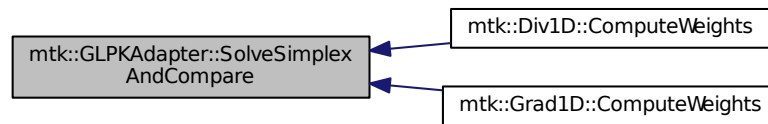
1. Memory allocation.
2. Fill the problem.
3. Copy the row to the vector objective.
4. Forming the RHS.
5. Setting up the objective function.
6. Setting up constraints.
7. Copy the matrix minus the row objective to the glpk problem.
8. Solve problem.

Definition at line 76 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:

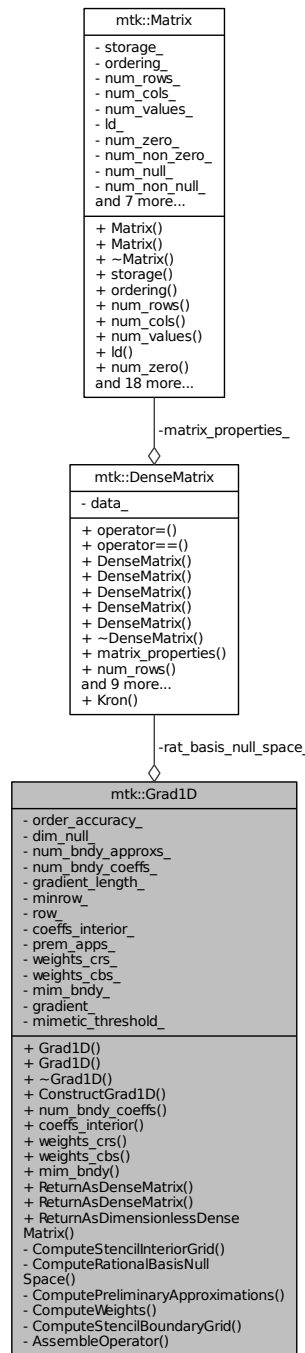
- [include/mtk_glpk_adapter.h](#)
- [src/mtk_glpk_adapter.cc](#)

16.8 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)
- *Returns the operator as a dense matrix.*
- `DenseMatrix ReturnAsDenseMatrix` (const `UniStgGrid1D` &grid)
- *Returns the operator as a dense matrix.*
- `DenseMatrix ReturnAsDimensionlessDenseMatrix` (int num_cells_x)
- *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.

16.8.1 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](#).

16.8.2 Constructor & Destructor Documentation

16.8.2.1 mtk::Grad1D::Grad1D ()

Definition at line 129 of file [mtk_grad_1d.cc](#).

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

Parameters

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

Definition at line 145 of file [mtk_grad_1d.cc](#).

16.8.2.3 mtk::Grad1D::~~Grad1D ()

Definition at line 161 of file [mtk_grad_1d.cc](#).

16.8.3 Member Function Documentation

16.8.3.1 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 1497 of file [mtk_grad_1d.cc](#).

16.8.3.2 mtk::Real * mtk::Grad1D::coeffs_interior () const

Returns

Coefficients for the interior of the grid.

Definition at line 330 of file [mtk_grad_1d.cc](#).

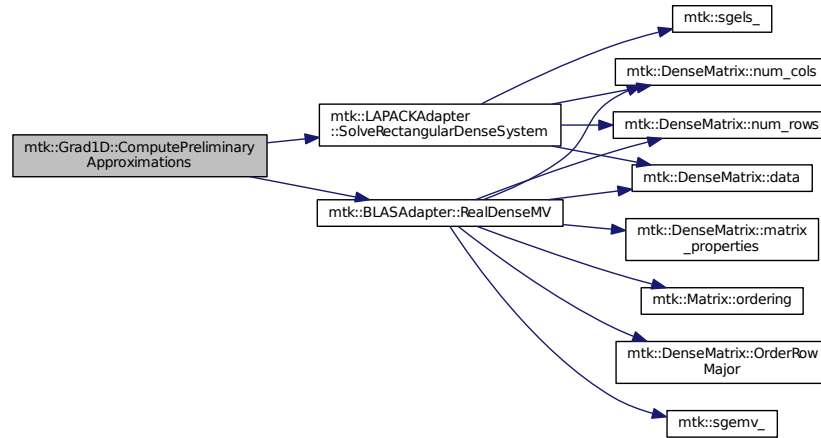
16.8.3.3 bool mtk::Grad1D::ComputePreliminaryApproximations (void) [private]

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 831 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



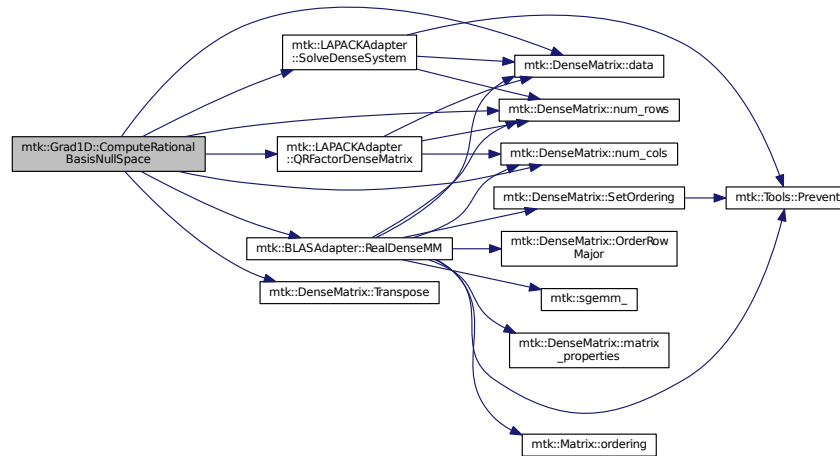
16.8.3.4 `bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) [private]`

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 648 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



16.8.3.5 `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 1391 of file [mtk_grad_1d.cc](#).

16.8.3.6 `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 552 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



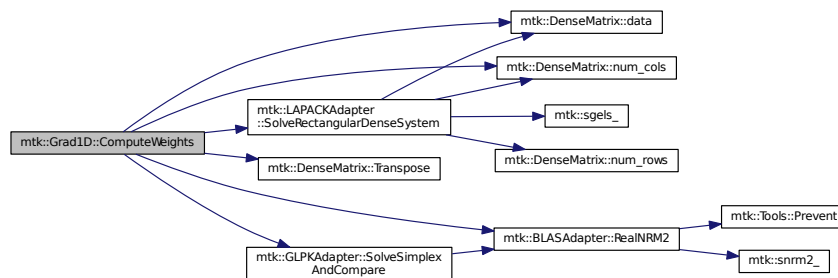
16.8.3.7 bool mtk::Grad1D::ComputeWeights (void) [private]

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 Φ matrix.
8. Apply solution found from brute force search.

Definition at line 1051 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



16.8.3.8 bool mtk::Grad1D::ConstructGrad1D (int order_accuracy = kDefaultOrderAccuracy, Real mimetic_threshold = kDefaultMimeticThreshold)

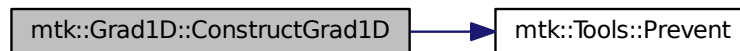
Returns

Success of the solution.

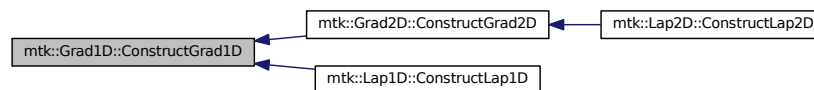
1. Compute stencil for the interior cells.
2. Compute a rational null-space from the first matrix transposed.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 182 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



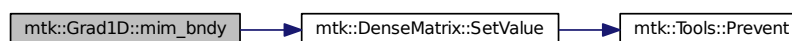
16.8.3.9 `mtk::DenseMatrix mtk::Grad1D::mim_bndy () const`

Returns

Collection of mimetic approximations at the boundary.

Definition at line 345 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



16.8.3.10 `int mtk::Grad1D::num_bndy_coeffs () const`

Returns

How many coefficients are approximating at the boundary.

Definition at line 325 of file [mtk_grad_1d.cc](#).

16.8.3.11 `mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (mtk::Real west, mtk::Real east, int num_cells_x)`

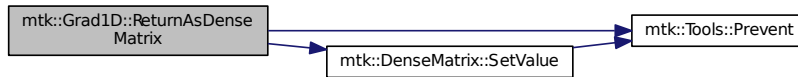
Returns

The operator as a dense matrix.

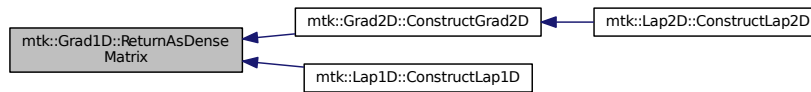
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 360 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.8.3.12 `mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid)`

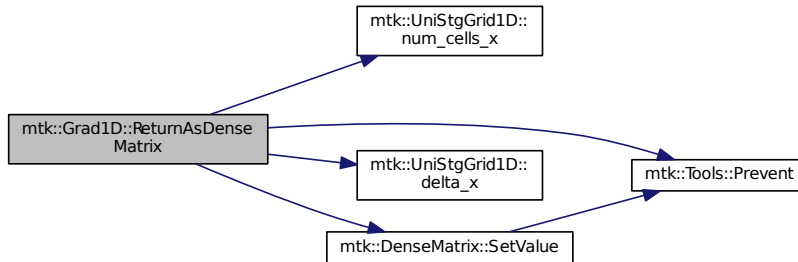
Returns

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 428 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



16.8.3.13 `mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix (int num_cells_x)`

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 491 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



16.8.3.14 `mtk::Real * mtk::Grad1D::weights_cbs (void) const`

Returns

Collection of weights as computed by the CBSA.

Definition at line 340 of file [mtk_grad_1d.cc](#).

16.8.3.15 `mtk::Real * mtk::Grad1D::weights_crs (void) const`

Returns

Success of the solution.

Definition at line 335 of file [mtk_grad_1d.cc](#).

16.8.4 Friends And Related Function Documentation

16.8.4.1 `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 79 of file [mtk_grad_1d.cc](#).

16.8.5 Member Data Documentation

16.8.5.1 `Real* mtk::Grad1D::coeffs_interior_` [*private*]

Definition at line 217 of file [mtk_grad_1d.h](#).

16.8.5.2 `int mtk::Grad1D::dim_null_` [*private*]

Definition at line 208 of file [mtk_grad_1d.h](#).

16.8.5.3 `Real* mtk::Grad1D::gradient_` [*private*]

Definition at line 222 of file [mtk_grad_1d.h](#).

16.8.5.4 `int mtk::Grad1D::gradient_length_` [*private*]

Definition at line 211 of file [mtk_grad_1d.h](#).

16.8.5.5 `Real* mtk::Grad1D::mim_bndy_` [*private*]

Definition at line 221 of file [mtk_grad_1d.h](#).

16.8.5.6 `Real mtk::Grad1D::mimetic_threshold_` [*private*]

Definition at line 224 of file [mtk_grad_1d.h](#).

16.8.5.7 `int mtk::Grad1D::minrow_ [private]`

Definition at line 212 of file [mtk_grad_1d.h](#).

16.8.5.8 `int mtk::Grad1D::num_bndy_approxs_ [private]`

Definition at line 209 of file [mtk_grad_1d.h](#).

16.8.5.9 `int mtk::Grad1D::num_bndy_coeffs_ [private]`

Definition at line 210 of file [mtk_grad_1d.h](#).

16.8.5.10 `int mtk::Grad1D::order_accuracy_ [private]`

Definition at line 207 of file [mtk_grad_1d.h](#).

16.8.5.11 `Real* mtk::Grad1D::prem_apps_ [private]`

Definition at line 218 of file [mtk_grad_1d.h](#).

16.8.5.12 `DenseMatrix mtk::Grad1D::rat_basis_null_space_ [private]`

Definition at line 215 of file [mtk_grad_1d.h](#).

16.8.5.13 `int mtk::Grad1D::row_ [private]`

Definition at line 213 of file [mtk_grad_1d.h](#).

16.8.5.14 `Real* mtk::Grad1D::weights_cbs_ [private]`

Definition at line 220 of file [mtk_grad_1d.h](#).

16.8.5.15 `Real* mtk::Grad1D::weights_crs_ [private]`

Definition at line 219 of file [mtk_grad_1d.h](#).

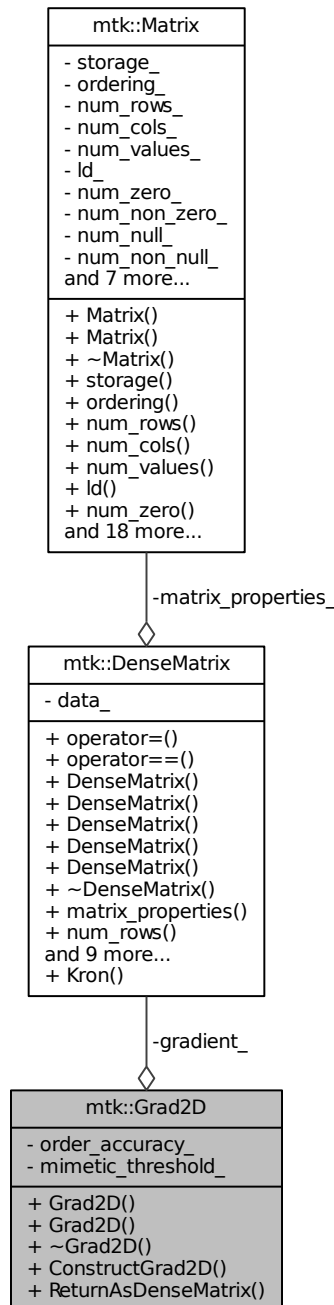
The documentation for this class was generated from the following files:

- [include/mtk_grad_1d.h](#)
- [src/mtk_grad_1d.cc](#)

16.9 mtk::Grad2D Class Reference

```
#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](#) ()

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) [gradient_](#)

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

- [Real](#) [mimetic_threshold_](#)

Mimetic Threshold.

16.9.1 Detailed Description

Definition at line 66 of file [mtk_grad_2d.h](#).

16.9.2 Constructor & Destructor Documentation

16.9.2.1 [mtk::Grad2D::Grad2D](#) ()

Definition at line 67 of file [mtk_grad_2d.cc](#).

16.9.2.2 [mtk::Grad2D::Grad2D](#) (const [Grad2D](#) &grad)

Parameters

in	div	Given divergence.
--------------------	---------------------	-------------------

Definition at line 71 of file [mtk_grad_2d.cc](#).

16.9.2.3 [mtk::Grad2D::~~Grad2D](#) ()

Definition at line 75 of file [mtk_grad_2d.cc](#).

16.9.3 Member Function Documentation

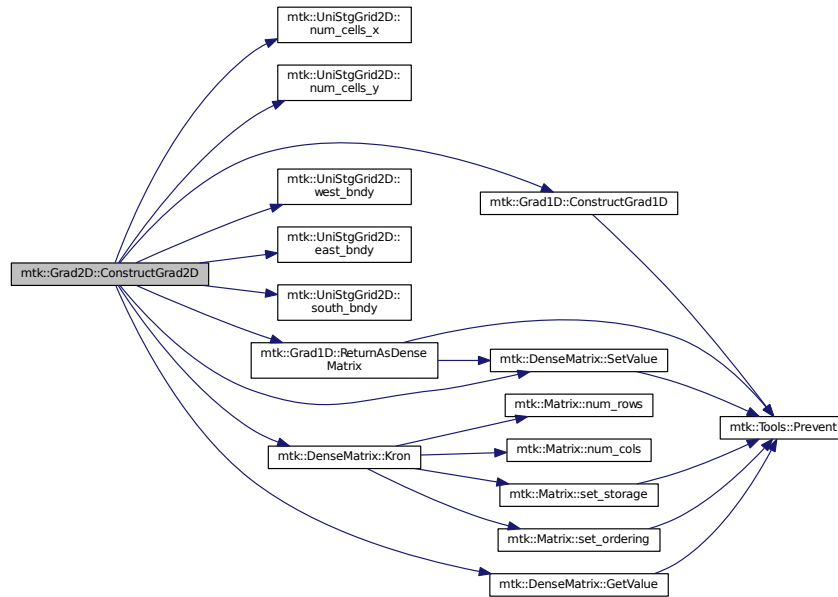
16.9.3.1 bool [mtk::Grad2D::ConstructGrad2D](#) (const [UniStgGrid2D](#) &grid, int order_accuracy = kDefaultOrderAccuracy, [mtk::Real](#) mimetic_threshold = kDefaultMimeticThreshold)

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:



16.9.3.2 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix ()

Returns

The operator as a dense matrix.

Definition at line 143 of file [mtk_grad_2d.cc](#).

Here is the caller graph for this function:

**16.9.4 Member Data Documentation****16.9.4.1 DenseMatrix mtk::Grad2D::gradient_ [private]**

Definition at line 98 of file [mtk_grad_2d.h](#).

16.9.4.2 Real mtk::Grad2D::mimetic_threshold_ [private]

Definition at line 102 of file [mtk_grad_2d.h](#).

16.9.4.3 int mtk::Grad2D::order_accuracy_ [private]

Definition at line 100 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](#)

16.10 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

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

Collaboration diagram for mtk::Interp1D:

mtk::Interp1D
<ul style="list-style-type: none"> - dir_interp_ - order_accuracy_ - coeffs_interior_
<ul style="list-style-type: none"> + Interp1D() + Interp1D() + ~Interp1D() + ConstructInterp1D() + coeffs_interior() + ReturnAsDenseMatrix()

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)
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.

16.10.1 Detailed Description

This class implements a 1D interpolation operator.

Definition at line 82 of file [mtk_interp_1d.h](#).

16.10.2 Constructor & Destructor Documentation

16.10.2.1 `mtk::Interp1D::Interp1D ()`

Definition at line 80 of file [mtk_interp_1d.cc](#).

16.10.2.2 `mtk::Interp1D::Interp1D (const Interp1D & interp)`

Parameters

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

Definition at line 85 of file [mtk_interp_1d.cc](#).

16.10.2.3 `mtk::Interp1D::~~Interp1D ()`

Definition at line 90 of file [mtk_interp_1d.cc](#).

16.10.3 Member Function Documentation

16.10.3.1 `mtk::Real * mtk::Interp1D::coeffs_interior () const`

Returns

Coefficients for the interior of the grid.

Definition at line 130 of file [mtk_interp_1d.cc](#).

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

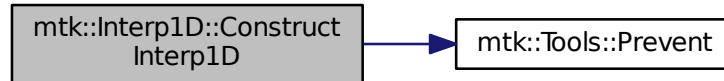
Returns

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:



16.10.3.3 mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid)

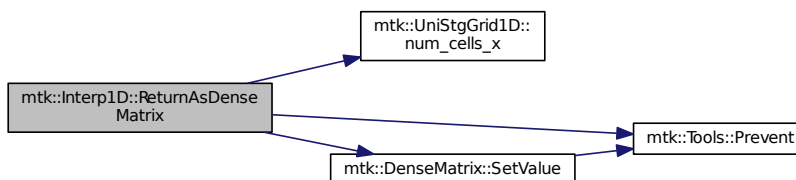
Returns

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 135 of file [mtk_interp_1d.cc](#).

Here is the call graph for this function:



16.10.4 Friends And Related Function Documentation

16.10.4.1 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](#).

16.10.5 Member Data Documentation

16.10.5.1 `Real* mtk::Interp1D::coeffs_interior_` `[private]`

Definition at line 127 of file [mtk_interp_1d.h](#).

16.10.5.2 `DirInterp mtk::Interp1D::dir_interp_` `[private]`

Definition at line 123 of file [mtk_interp_1d.h](#).

16.10.5.3 `int mtk::Interp1D::order_accuracy_` `[private]`

Definition at line 125 of file [mtk_interp_1d.h](#).

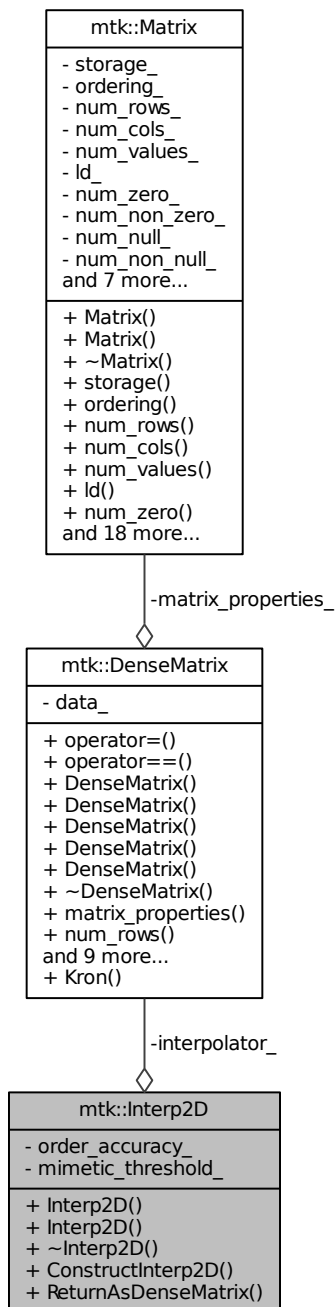
The documentation for this class was generated from the following files:

- [include/mtk_interp_1d.h](#)
- [src/mtk_interp_1d.cc](#)

16.11 mtk::Interp2D Class Reference

```
#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.

16.11.1 Detailed Description

Definition at line 67 of file [mtk_interp_2d.h](#).

16.11.2 Constructor & Destructor Documentation

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

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

Parameters

in	lap	Given Laplacian.
--------------------	---------------------	------------------

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

16.11.3 Member Function Documentation

16.11.3.1 [DenseMatrix mtk::Interp2D::ConstructInterp2D](#) (const [UniStgGrid2D](#) & *grid*, int *order_accuracy* = [kDefaultOrderAccuracy](#), [Real](#) *mimetic_threshold* = [kDefaultMimeticThreshold](#))

Returns

Success of the construction.

16.11.3.2 DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix ()

Returns

The operator as a dense matrix.

16.11.4 Member Data Documentation

16.11.4.1 DenseMatrix mtk::Interp2D::interpolator_ [private]

Definition at line 99 of file [mtk_interp_2d.h](#).

16.11.4.2 Real mtk::Interp2D::mimetic_threshold_ [private]

Definition at line 103 of file [mtk_interp_2d.h](#).

16.11.4.3 int mtk::Interp2D::order_accuracy_ [private]

Definition at line 101 of file [mtk_interp_2d.h](#).

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

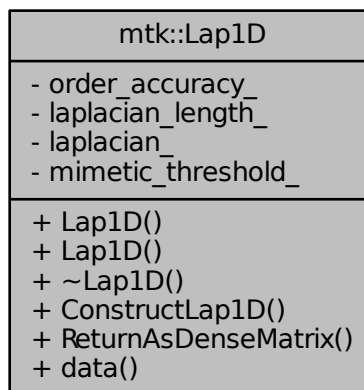
- [include/mtk_interp_2d.h](#)

16.12 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.
- bool [ConstructLap1D](#) (int order_accuracy=kDefaultOrderAccuracy, [Real](#) mimetic_threshold=kDefaultMimeticThreshold)
Factory method implementing the CBS Algorithm to build operator.
- [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid)
Return the operator as a dense matrix.
- [mtk::Real](#) * [data](#) (const [UniStgGrid1D](#) &grid)
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](#) [mimetic_threshold_](#)
< Mimetic threshold.

Friends

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

16.12.1 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](#).

16.12.2 Constructor & Destructor Documentation

16.12.2.1 [mtk::Lap1D::Lap1D](#) ()

Definition at line 108 of file [mtk_lap_1d.cc](#).

16.12.2.2 [mtk::Lap1D::Lap1D](#) (const [Lap1D](#) & lap)

Parameters

<i>in</i>	<i>lap</i>	Given Laplacian.
-----------	------------	------------------

16.12.2.3 mtk::Lap1D::~~Lap1D ()

Definition at line 113 of file [mtk_lap_1d.cc](#).

16.12.3 Member Function Documentation

16.12.3.1 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.

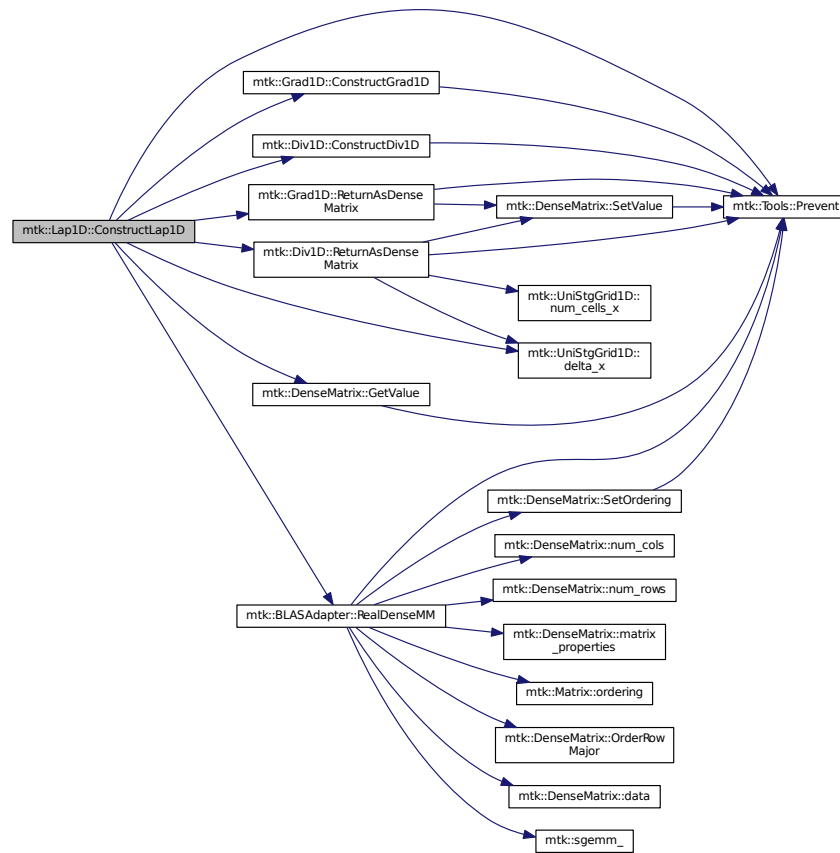
Warning

We do not compute weights for this operator.

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 119 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



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

Returns

The operator as a dense array.

Definition at line 332 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



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

Returns

The operator as a dense matrix.

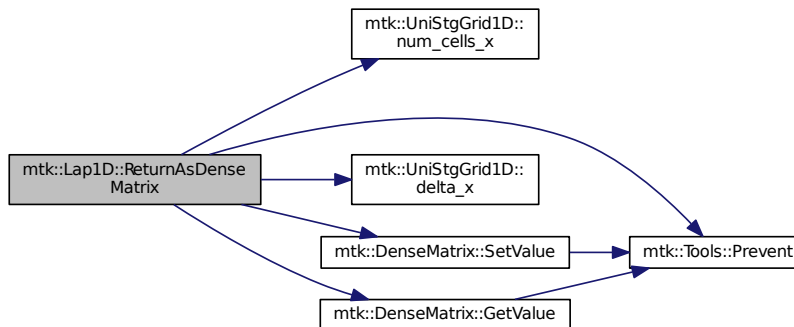
1. Extract mimetic coefficients from the west boundary.
2. Extract interior coefficients.
3. Extract mimetic coefficients from the west boundary to go east.

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 265 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



16.12.4 Friends And Related Function Documentation

16.12.4.1 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](#).

16.12.5 Member Data Documentation

16.12.5.1 Real* mtk::Lap1D::laplacian_ [private]

Definition at line 120 of file [mtk_lap_1d.h](#).

16.12.5.2 `int mtk::Lap1D::laplacian_length_ [private]`

Definition at line [118](#) of file [mtk_lap_1d.h](#).

16.12.5.3 `Real mtk::Lap1D::mimetic_threshold_ [private]`

Definition at line [122](#) of file [mtk_lap_1d.h](#).

16.12.5.4 `int mtk::Lap1D::order_accuracy_ [private]`

Definition at line [117](#) of file [mtk_lap_1d.h](#).

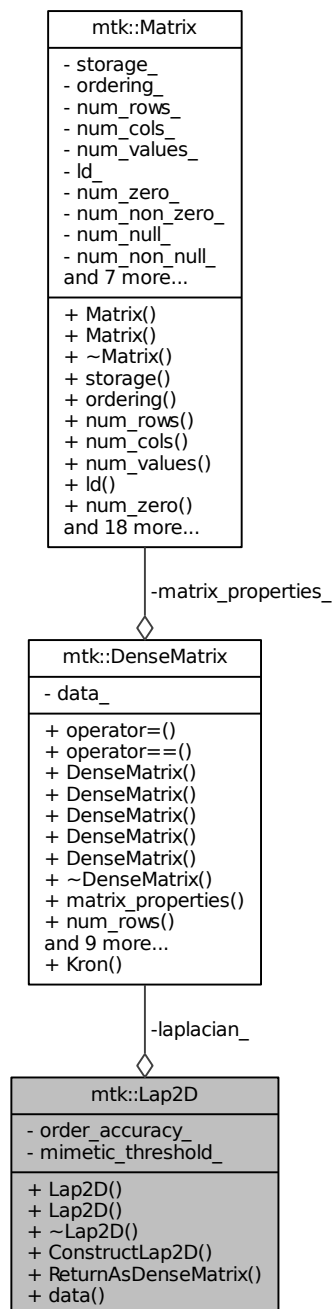
The documentation for this class was generated from the following files:

- [include/mtk_lap_1d.h](#)
- [src/mtk_lap_1d.cc](#)

16.13 mtk::Lap2D Class Reference

```
#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](#) ()

Return the operator as a dense matrix.

- [mtk::Real](#) * [data](#) ()

Return the operator as a dense array.

Private Attributes

- [DenseMatrix](#) [laplacian_](#)

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

- [Real](#) [mimetic_threshold_](#)

Mimetic Threshold.

16.13.1 Detailed Description

Definition at line 66 of file [mtk_lap_2d.h](#).

16.13.2 Constructor & Destructor Documentation

16.13.2.1 [mtk::Lap2D::Lap2D](#) ()

Definition at line 69 of file [mtk_lap_2d.cc](#).

16.13.2.2 [mtk::Lap2D::Lap2D](#) (const [Lap2D](#) & *lap*)

Parameters

<i>in</i>	<i>lap</i>	Given Laplacian.
-----------	------------	------------------

Definition at line 71 of file [mtk_lap_2d.cc](#).

16.13.2.3 [mtk::Lap2D::~~Lap2D](#) ()

Definition at line 75 of file [mtk_lap_2d.cc](#).

16.13.3 Member Function Documentation

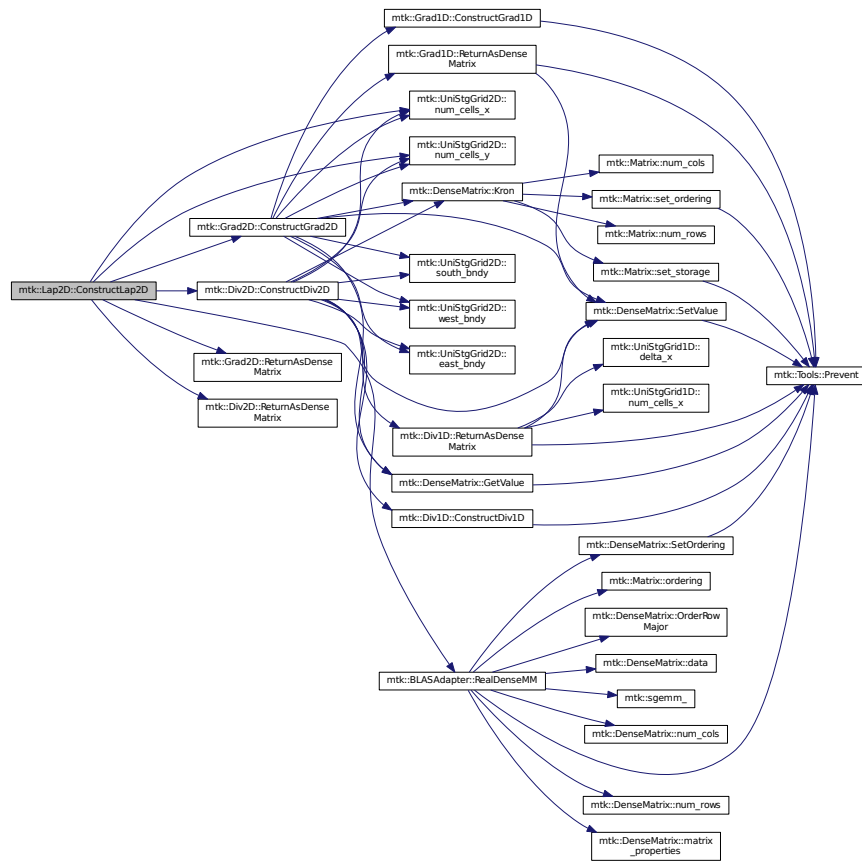
16.13.3.1 `bool mtk::Lap2D::ConstructLap2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

Definition at line 77 of file [mtk_lap_2d.cc](#).

Here is the call graph for this function:



16.13.3.2 `mtk::Real * mtk::Lap2D::data ()`

Returns

The operator as a dense array.

Definition at line 115 of file [mtk_lap_2d.cc](#).

16.13.3.3 `mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix ()`

Returns

The operator as a dense matrix.

Definition at line 110 of file [mtk_lap_2d.cc](#).

16.13.4 Member Data Documentation**16.13.4.1 DenseMatrix mtk::Lap2D::laplacian_ [private]**

Definition at line 105 of file [mtk_lap_2d.h](#).

16.13.4.2 Real mtk::Lap2D::mimetic_threshold_ [private]

Definition at line 109 of file [mtk_lap_2d.h](#).

16.13.4.3 int mtk::Lap2D::order_accuracy_ [private]

Definition at line 107 of file [mtk_lap_2d.h](#).

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

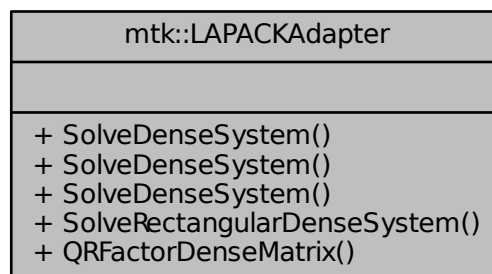
- [include/mtk_lap_2d.h](#)
- [src/mtk_lap_2d.cc](#)

16.14 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 [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.

16.14.1 Detailed Description

This class contains a collection of static classes, 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 92 of file [mtk_lapack_adapter.h](#).

16.14.2 Member Function Documentation

16.14.2.1 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix (mtk::DenseMatrix & aa) [static]

Adapts the MTK to LAPACK's routine.

Parameters

<i>in, out</i>	<i>matrix</i>	Input matrix.
----------------	---------------	---------------

Returns

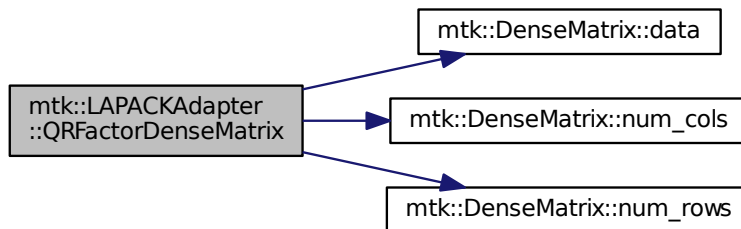
[Matrix Q](#).

Exceptions

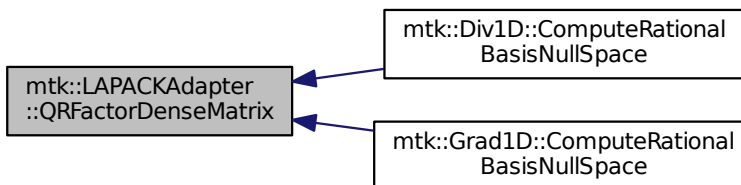
<i>std::bad_alloc</i>

Definition at line 555 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.2.2 `int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & mm, mtk::Real * rhs) [static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

Parameters

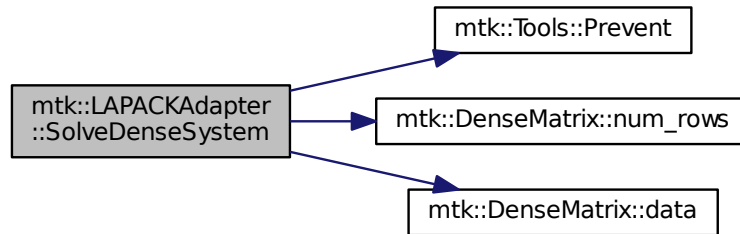
<code>in</code>	<code>matrix</code>	Input matrix.
<code>in</code>	<code>rhs</code>	Input right-hand sides vector.

Exceptions

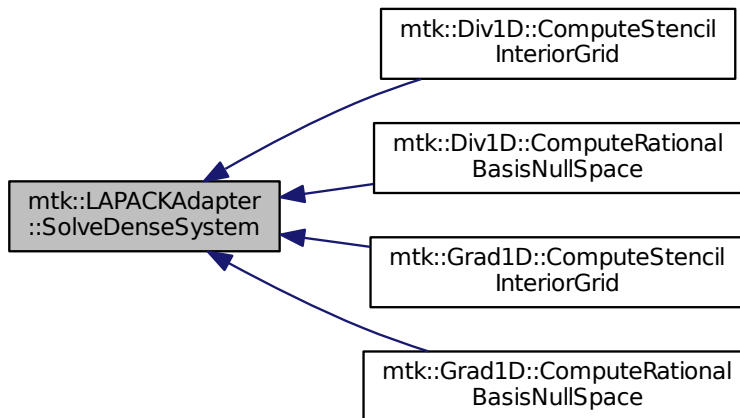
<code>std::bad_alloc</code>

Definition at line [430](#) of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.2.3 `int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & mm, mtk::DenseMatrix & rr) [static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

Parameters

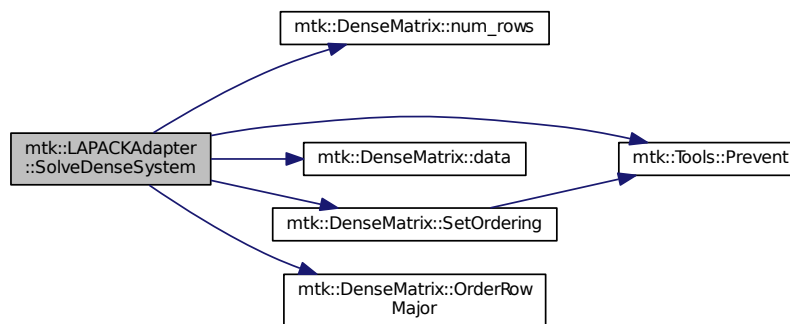
<code>in</code>	<code>matrix</code>	Input matrix.
<code>in</code>	<code>rr</code>	Input right-hand sides matrix.

Exceptions

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

Definition at line 465 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



16.14.2.4 `int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & mm, mtk::UniStgGrid1D & rhs)`
`[static]`

Adapts the MTK to LAPACK's `dgesv_` routine.

Parameters

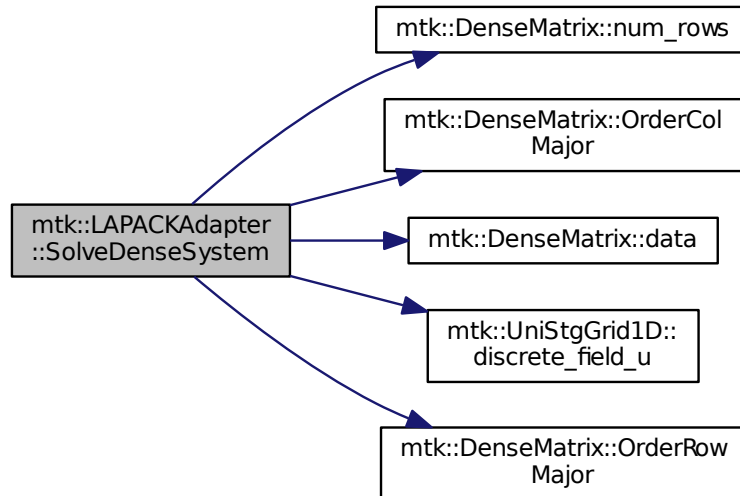
in	<i>matrix</i>	Input matrix.
in	<i>rr</i>	Input right-hand side from info on a grid.

Exceptions

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

Definition at line 517 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



16.14.2.5 `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

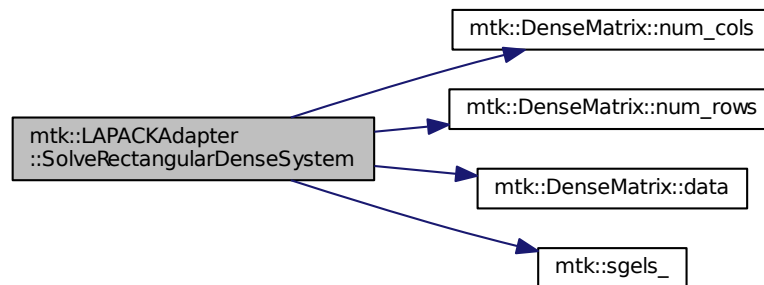
Success of the solution.

Exceptions

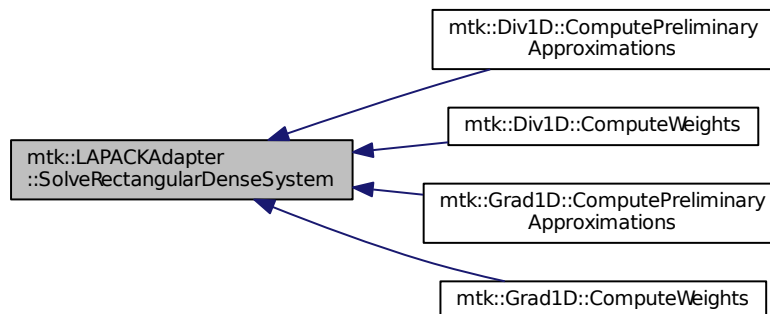
<code>std::bad_alloc</code>	
-----------------------------	--

Definition at line 756 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](#)

16.15 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"> - storage_ - ordering_ - num_rows_ - num_cols_ - num_values_ - ld_ - num_zero_ - num_non_zero_ - num_null_ - num_non_null_ and 7 more...
<ul style="list-style-type: none"> + Matrix() + Matrix() + ~Matrix() + storage() + ordering() + num_rows() + num_cols() + num_values() + ld() + num_zero() and 18 more...

Public Member Functions

- [Matrix](#) ()
Default constructor.
- [Matrix](#) (const [Matrix](#) &in)
Copy constructor.
- [~Matrix](#) ()
Destructor.
- [MatrixStorage](#) storage () const
Gets the type of storage of this matrix.
- [MatrixOrdering](#) ordering () const
Gets the type of ordering of this matrix.
- int [num_rows](#) () const
Gets the number of rows.
- int [num_cols](#) () const
Gets the number of rows.

- `int num_values () const`
Gets the number of values.
- `int ld () const`
Gets the matrix' leading dimension.
- `int num_zero () const`
Gets the number of zeros.
- `int num_non_zero () const`
Gets the number of non-zero values.
- `int num_null () const`
Gets the number of null values.
- `int num_non_null () const`
Gets the number of non-null values.
- `int kl () const`
Gets the number of lower diagonals.
- `int ku () const`
Gets the number of upper diagonals.
- `int bandwidth () const`
Gets the bandwidth.
- `Real abs_density () const`
Gets the absolute density.
- `Real rel_density () const`
Gets the relative density.
- `Real abs_sparsity () const`
Gets the Absolute sparsity.
- `Real rel_sparsity () const`
Gets the Relative sparsity.
- `void set_storage (const MatrixStorage &tt)`
Sets the storage type of the matrix.
- `void set_ordering (const MatrixOrdering &oo)`
Sets the ordering of the matrix.
- `void set_num_rows (int num_rows)`
Sets the number of rows of the matrix.
- `void set_num_cols (int num_cols)`
Sets the number of columns of the matrix.
- `void set_num_zero (int in)`
Sets the number of zero values of the matrix that matter.
- `void set_num_null (int in)`
Sets the number of zero values of the matrix that DO NOT matter.
- `void IncreaseNumZero ()`
Increases the number of values that equal zero but with meaning.
- `void IncreaseNumNull ()`
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.

16.15.1 Detailed Description

Definition of the representation for the matrices implemented in the MTK.

Definition at line 75 of file [mtk_matrix.h](#).

16.15.2 Constructor & Destructor Documentation

16.15.2.1 mtk::Matrix::Matrix ()

Definition at line 67 of file [mtk_matrix.cc](#).

16.15.2.2 mtk::Matrix::Matrix (const Matrix & *in*)

Parameters

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

Definition at line 86 of file [mtk_matrix.cc](#).

16.15.2.3 mtk::Matrix::~~Matrix ()

Definition at line 105 of file [mtk_matrix.cc](#).

16.15.3 Member Function Documentation

16.15.3.1 Real mtk::Matrix::abs_density () const

See also

http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf

Returns

Absolute density of the matrix.

16.15.3.2 mtk::Real mtk::Matrix::abs_sparsity () const

See also

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](#).

16.15.3.3 int mtk::Matrix::bandwidth () const

Returns

Bandwidth of the matrix.

Definition at line 167 of file [mtk_matrix.cc](#).

16.15.3.4 void mtk::Matrix::IncreaseNumNull ()

Todo Review the definition of sparse matrices properties.

Definition at line 274 of file [mtk_matrix.cc](#).

16.15.3.5 void mtk::Matrix::IncreaseNumZero ()

Todo Review the definition of sparse matrices properties.

Definition at line 264 of file [mtk_matrix.cc](#).

16.15.3.6 int mtk::Matrix::kl () const

Returns

Number of lower diagonals.

Definition at line 157 of file [mtk_matrix.cc](#).

16.15.3.7 int mtk::Matrix::ku () const

Returns

Number of upper diagonals.

Definition at line 162 of file [mtk_matrix.cc](#).

16.15.3.8 int mtk::Matrix::ld () const

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](#).

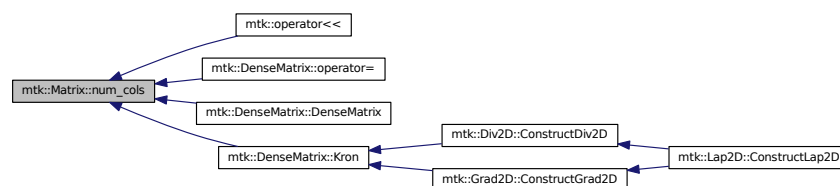
16.15.3.9 int mtk::Matrix::num_cols () const

Returns

Number of rows of the matrix.

Definition at line 122 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



16.15.3.10 `int mtk::Matrix::num_non_null () const`

See also

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`.

16.15.3.11 `int mtk::Matrix::num_non_zero () const`

Returns

Number of non-zero values of the matrix.

Definition at line 142 of file `mtk_matrix.cc`.

16.15.3.12 `int mtk::Matrix::num_null () const`

See also

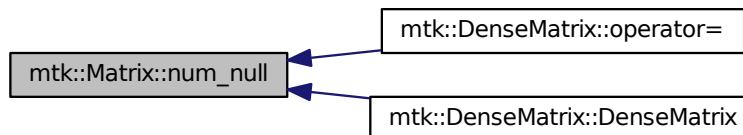
http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf

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:



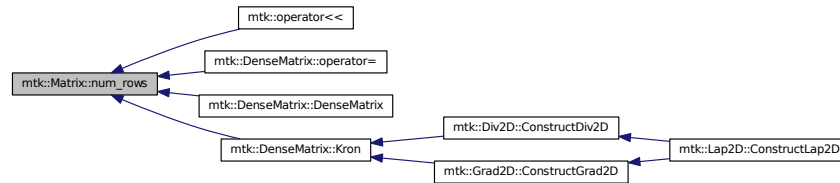
16.15.3.13 `int mtk::Matrix::num_rows () const`

Returns

Number of rows of the matrix.

Definition at line 117 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



16.15.3.14 int mtk::Matrix::num_values () const

Returns

Number of values of the matrix.

Definition at line 127 of file [mtk_matrix.cc](#).

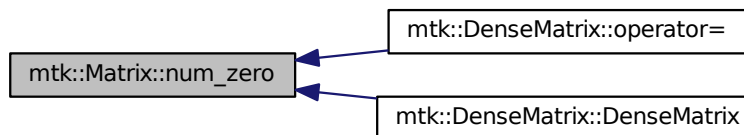
16.15.3.15 int mtk::Matrix::num_zero () const

Returns

Number of zeros of the matrix.

Definition at line 137 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



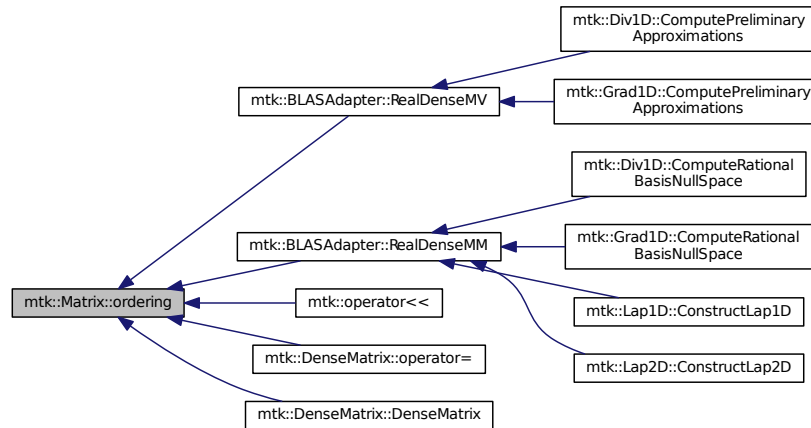
16.15.3.16 mtk::MatrixOrdering mtk::Matrix::ordering () const

Returns

Type of ordering of this matrix.

Definition at line 112 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



16.15.3.17 mtk::Real mtk::Matrix::rel_density () const

See also

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](#).

16.15.3.18 mtk::Real mtk::Matrix::rel_sparsity () const

See also

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](#).

16.15.3.19 void mtk::Matrix::set_num_cols (int num_cols)

Parameters

<i>in</i>	<i>num_cols</i>	Number of columns.
-----------	-----------------	--------------------

Definition at line 224 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.15.3.20 void `mtk::Matrix::set_num_null` (int *in*)

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 250 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.15.3.21 `void mtk::Matrix::set_num_rows (int num_rows)`

Parameters

<i>in</i>	<i>num_rows</i>	Number of rows.
-----------	-----------------	-----------------

Definition at line 212 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.15.3.22 `void mtk::Matrix::set_num_zero (int in)`

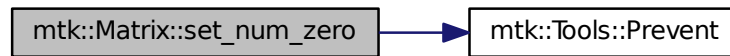
Parameters

<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 236 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.15.3.23 void mtk::Matrix::set_ordering (const MatrixOrdering & oo)

See also

[MatrixOrdering](#)

Parameters

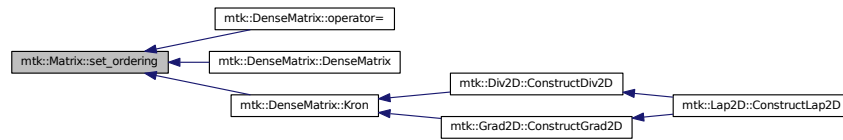
in	oo	Ordering of the matrix.
----	----	-------------------------

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:



16.15.3.24 void mtk::Matrix::set_storage (const MatrixStorage & tt)

See also

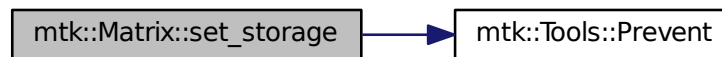
[MatrixStorage](#)

Parameters

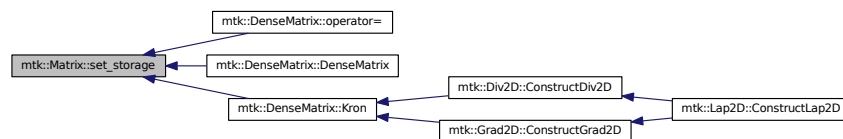
in	tt	Type of the matrix storage.
----	----	-----------------------------

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:



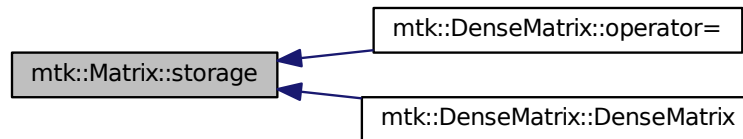
16.15.3.25 mtk::MatrixStorage mtk::Matrix::storage () const

Returns

Type of storage of this matrix.

Definition at line 107 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:

**16.15.4 Member Data Documentation****16.15.4.1 Real mtk::Matrix::abs_density_ [private]**

Definition at line 296 of file [mtk_matrix.h](#).

16.15.4.2 Real mtk::Matrix::abs_sparsity_ [private]

Definition at line 298 of file [mtk_matrix.h](#).

16.15.4.3 int mtk::Matrix::bandwidth_ [private]

Definition at line 294 of file [mtk_matrix.h](#).

16.15.4.4 int mtk::Matrix::kl_ [private]

Definition at line 292 of file [mtk_matrix.h](#).

16.15.4.5 int mtk::Matrix::ku_ [private]

Definition at line 293 of file [mtk_matrix.h](#).

16.15.4.6 int mtk::Matrix::ld_ [private]

Definition at line 285 of file [mtk_matrix.h](#).

16.15.4.7 int mtk::Matrix::num_cols_ [private]

Definition at line 283 of file [mtk_matrix.h](#).

16.15.4.8 `int mtk::Matrix::num_non_null_` [private]

Definition at line 290 of file [mtk_matrix.h](#).

16.15.4.9 `int mtk::Matrix::num_non_zero_` [private]

Definition at line 288 of file [mtk_matrix.h](#).

16.15.4.10 `int mtk::Matrix::num_null_` [private]

Definition at line 289 of file [mtk_matrix.h](#).

16.15.4.11 `int mtk::Matrix::num_rows_` [private]

Definition at line 282 of file [mtk_matrix.h](#).

16.15.4.12 `int mtk::Matrix::num_values_` [private]

Definition at line 284 of file [mtk_matrix.h](#).

16.15.4.13 `int mtk::Matrix::num_zero_` [private]

Definition at line 287 of file [mtk_matrix.h](#).

16.15.4.14 **MatrixOrdering** `mtk::Matrix::ordering_` [private]

Definition at line 280 of file [mtk_matrix.h](#).

16.15.4.15 **Real** `mtk::Matrix::rel_density_` [private]

Definition at line 297 of file [mtk_matrix.h](#).

16.15.4.16 **Real** `mtk::Matrix::rel_sparsity_` [private]

Definition at line 299 of file [mtk_matrix.h](#).

16.15.4.17 **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:

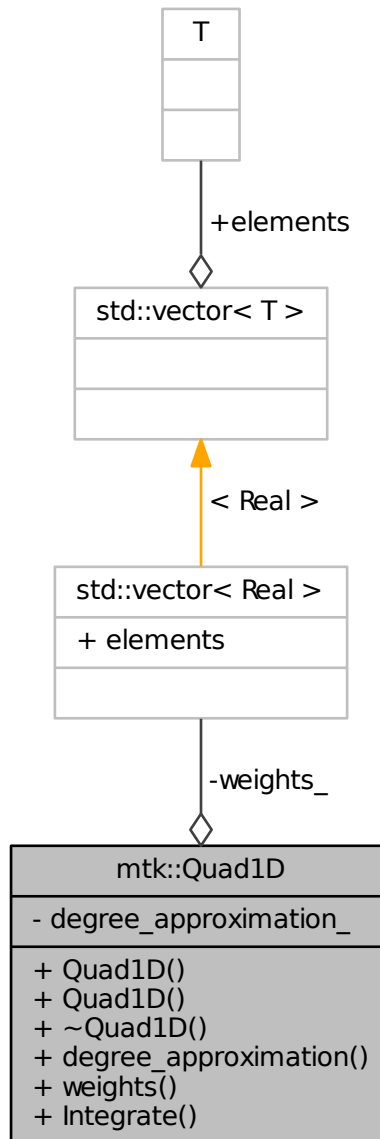
- [include/mtk_matrix.h](#)
- [src/mtk_matrix.cc](#)

16.16 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)
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.

16.16.1 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](#).

16.16.2 Constructor & Destructor Documentation

16.16.2.1 [mtk::Quad1D::Quad1D](#) ()

16.16.2.2 [mtk::Quad1D::Quad1D](#) (const [Quad1D](#) & quad)

Parameters

in	div	Given quadrature.
--------------------	---------------------	-------------------

16.16.2.3 `mtk::Quad1D::~~Quad1D ()`

16.16.3 Member Function Documentation

16.16.3.1 `int mtk::Quad1D::degree_approximation () const`

Returns

Degree of the interpolating polynomial per sub-interval of the domain.

16.16.3.2 `Real mtk::Quad1D::Integrate (Real(*) (Real xx) Integrand, UniStgGrid1D grid)`

Parameters

<code>in</code>	<i>Integrand</i>	Real-valued function to integrate.
<code>in</code>	<i>grid</i>	Given integration domain.

Returns

Result of the integration.

16.16.3.3 `Real* mtk::Quad1D::weights () const`

Returns

Collection of weights.

16.16.4 Friends And Related Function Documentation

16.16.4.1 `std::ostream& operator<< (std::ostream & stream, Quad1D & in)` [`friend`]

16.16.5 Member Data Documentation

16.16.5.1 `int mtk::Quad1D::degree_approximation_` [`private`]

Definition at line 124 of file [mtk_quad_1d.h](#).

16.16.5.2 `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:

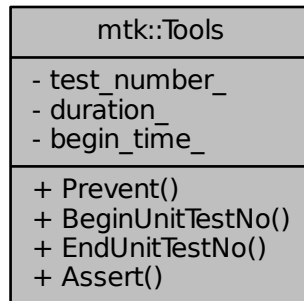
- [include/mtk_quad_1d.h](#)

16.17 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 *fname, int lineno, const char *fxname)
Enforces preconditions by preventing their complements from occur.
- static void [BeginUnitTestNo](#) (const int &nn)
Begins the execution of a unit test. Starts a timer.
- static void [EndUnitTestNo](#) (const int &nn)
Ends the execution of a unit test. Stops and reports wall-clock time.
- static void [Assert](#) (const bool condition)
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.

16.17.1 Detailed Description

Basic tools to ensure execution correctness.

Definition at line 74 of file [mtk_tools.h](#).

16.17.2 Member Function Documentation

16.17.2.1 void mtk::Tools::Assert (const bool *condition*) [static]

Parameters

<i>in</i>	<i>condition</i>	Condition to be asserted.
-----------	------------------	---------------------------

Definition at line 114 of file [mtk_tools.cc](#).

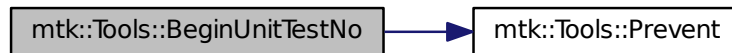
16.17.2.2 `void mtk::Tools::BeginUnitTestNo (const int & nn) [static]`

Parameters

<i>in</i>	<i>nn</i>	Number of the test.
-----------	-----------	---------------------

Definition at line 91 of file [mtk_tools.cc](#).

Here is the call graph for this function:



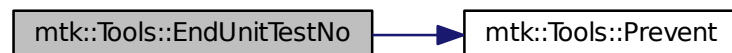
16.17.2.3 `void mtk::Tools::EndUnitTestNo (const int & nn) [static]`

Parameters

<i>in</i>	<i>nn</i>	Number of the test.
-----------	-----------	---------------------

Definition at line 105 of file [mtk_tools.cc](#).

Here is the call graph for this function:



16.17.2.4 `void mtk::Tools::Prevent (const bool complement, const char * fname, int lineno, const char * fxname) [static]`

See also

<http://stackoverflow.com/questions/8884335/print-the-file-name-line-number-and-function>

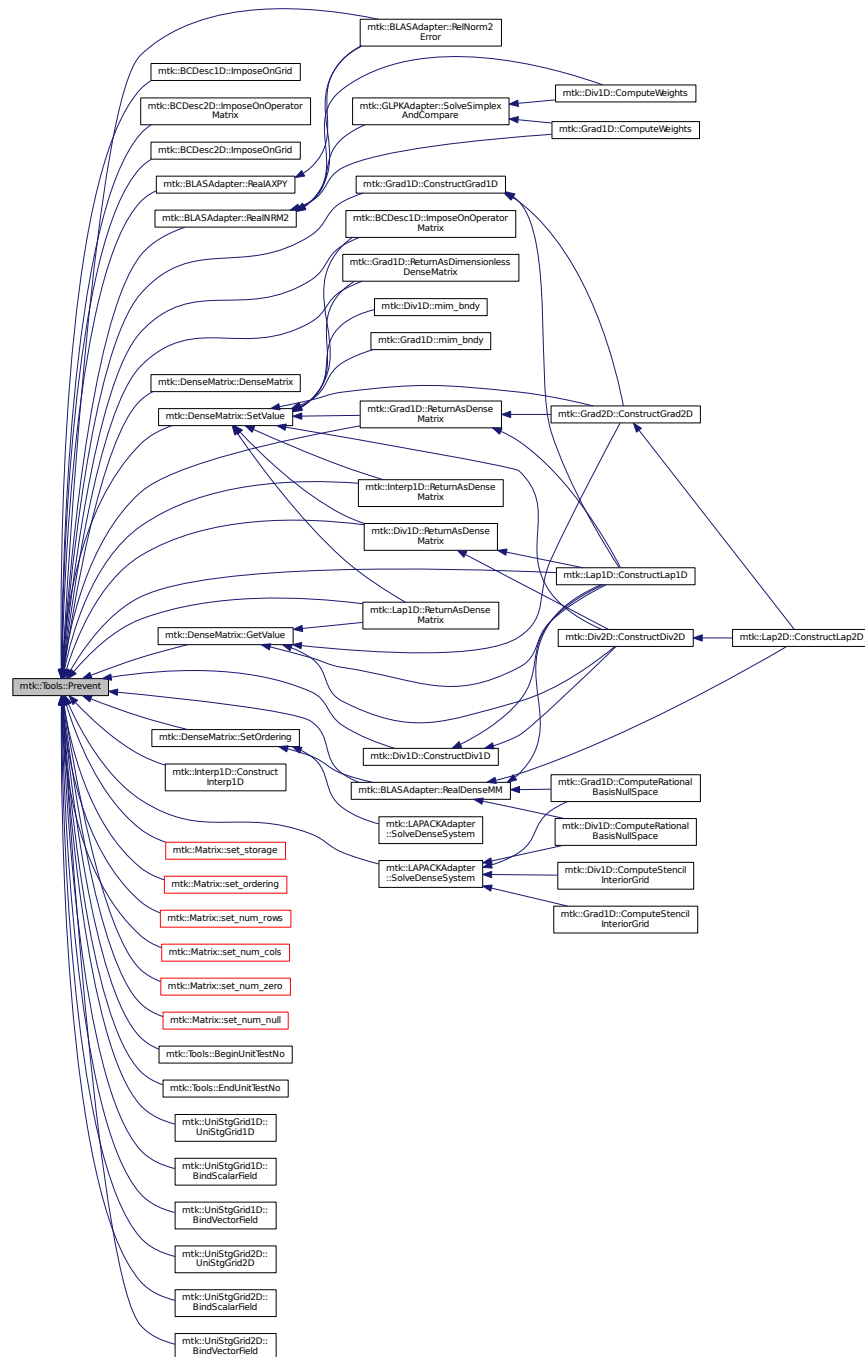
Parameters

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 61 of file [mtk_tools.cc](#).

Here is the caller graph for this function:



16.17.3 Member Data Documentation

16.17.3.1 clock_t mtk::Tools::begin_time_ [static], [private]

Definition at line 117 of file [mtk_tools.h](#).

16.17.3.2 `mtk::Real mtk::Tools::duration_` `[static], [private]`

Definition at line 115 of file [mtk_tools.h](#).

16.17.3.3 `int mtk::Tools::test_number_` `[static], [private]`

Todo Check usage of static methods and private members.

Definition at line 113 of file [mtk_tools.h](#).

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

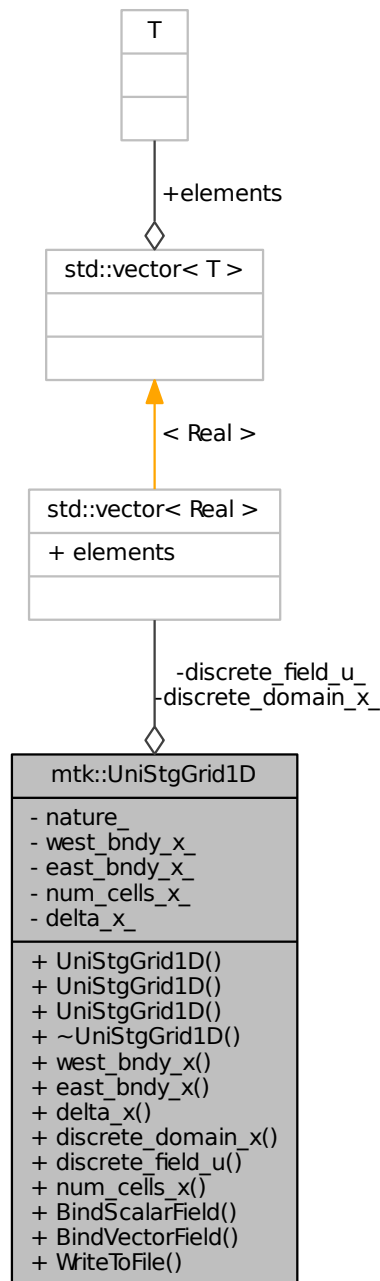
- [include/mtk_tools.h](#)
- [src/mtk_tools.cc](#)

16.18 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 &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 Δx .

- [Real](#) * discrete_domain_x ()

Provides access to the grid spatial data.

- [Real](#) * discrete_field_u ()

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)([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)

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_u_

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 Δx .

Friends

- std::ostream & operator<< (std::ostream &stream, [UniStgGrid1D](#) &in)

Prints the grid as a tuple of arrays.

16.18.1 Detailed Description

Uniform 1D Staggered Grid.

Definition at line 77 of file [mtk_uni_stg_grid_1d.h](#).

16.18.2 Constructor & Destructor Documentation

16.18.2.1 `mtk::UniStgGrid1D::UniStgGrid1D ()`

Definition at line 99 of file [mtk_uni_stg_grid_1d.cc](#).

16.18.2.2 `mtk::UniStgGrid1D::UniStgGrid1D (const UniStgGrid1D & grid)`

Parameters

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

Definition at line 108 of file [mtk_uni_stg_grid_1d.cc](#).

16.18.2.3 `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

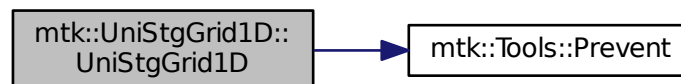
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>nature</i>	Nature of the discrete field to hold.

See also

[mtk::FieldNature](#)

Definition at line 124 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



16.18.2.4 `mtk::UniStgGrid1D::~~UniStgGrid1D ()`

Definition at line 144 of file [mtk_uni_stg_grid_1d.cc](#).

16.18.3 Member Function Documentation

16.18.3.1 void mtk::UniStgGrid1D::BindScalarField (*Real*(*)(*Real* xx) *ScalarField*)

Parameters

in	<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:



16.18.3.2 void mtk::UniStgGrid1D::BindVectorField (*Real*(*)(*Real* xx) *VectorField*)

We assume the field to be of the form:

$$\mathbf{v}(x) = v(x)\hat{\mathbf{i}}$$

Parameters

in	<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:



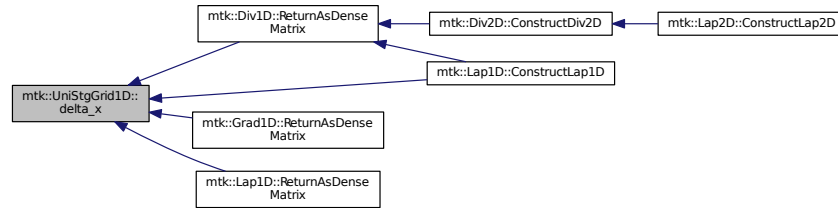
16.18.3.3 `mtk::Real mtk::UniStgGrid1D::delta_x () const`

Returns

Computed Δx .

Definition at line 156 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



16.18.3.4 `mtk::Real * mtk::UniStgGrid1D::discrete_domain_x ()`

Returns

Pointer to the spatial data.

Definition at line 161 of file [mtk_uni_stg_grid_1d.cc](#).

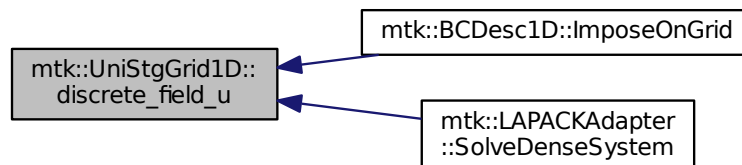
16.18.3.5 `mtk::Real * mtk::UniStgGrid1D::discrete_field_u ()`

Returns

Pointer to the field data.

Definition at line 166 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



16.18.3.6 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](#).

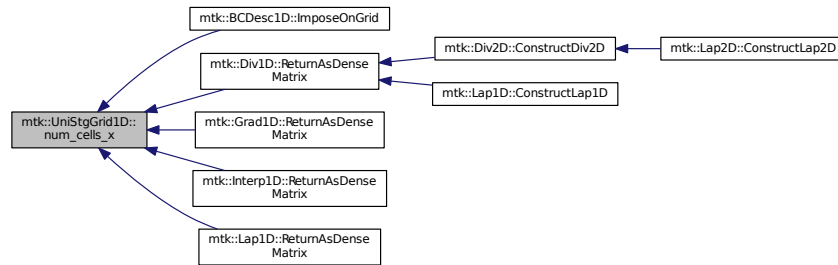
16.18.3.7 int mtk::UniStgGrid1D::num_cells_x () const

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:



16.18.3.8 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](#).

16.18.3.9 bool mtk::UniStgGrid1D::WriteToFile (std::string filename, std::string space_name, std::string field_name)

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](#).

16.18.4 Friends And Related Function Documentation

16.18.4.1 `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](#).

16.18.5 Member Data Documentation

16.18.5.1 `Real mtk::UniStgGrid1D::delta_x_` [private]

Definition at line 196 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.2 `std::vector<Real> mtk::UniStgGrid1D::discrete_domain_x_` [private]

Definition at line 190 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.3 `std::vector<Real> mtk::UniStgGrid1D::discrete_field_u_` [private]

Definition at line 191 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.4 `Real mtk::UniStgGrid1D::east_bndy_x_` [private]

Definition at line 194 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.5 `FieldNature mtk::UniStgGrid1D::nature_` [private]

Definition at line 188 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.6 `Real mtk::UniStgGrid1D::num_cells_x_` [private]

Definition at line 195 of file [mtk_uni_stg_grid_1d.h](#).

16.18.5.7 `Real mtk::UniStgGrid1D::west_bndy_x_` [private]

Definition at line 193 of file [mtk_uni_stg_grid_1d.h](#).

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

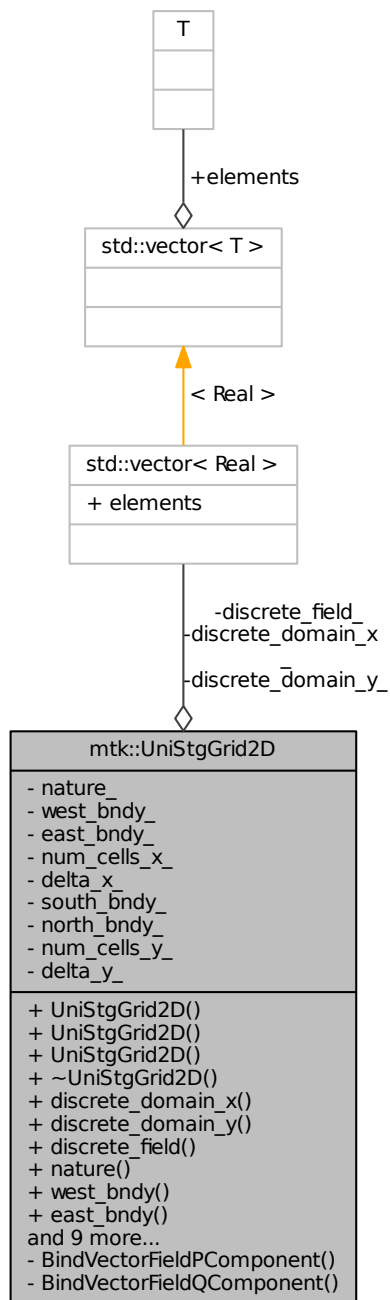
- [include/mtk_uni_stg_grid_1d.h](#)
- [src/mtk_uni_stg_grid_1d.cc](#)

16.19 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.
- [Real](#) * [discrete_domain_x](#) ()
Provides access to the grid spatial data.
- [Real](#) * [discrete_domain_y](#) ()
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 Δx .
- [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 Δy .
- void [BindScalarField](#) ([Real](#)(*ScalarField)([Real](#) xx, [Real](#) yy))
Binds a given scalar field to the grid.
- void [BindVectorField](#) ([Real](#)(*VectorFieldPComponent)([Real](#) xx, [Real](#) yy), [Real](#)(*VectorFieldQComponent)([Real](#) xx, [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)
Writes grid to a file compatible with Gnuplot 4.6.

Private Member Functions

- void [BindVectorFieldPComponent](#) ([Real](#)(*VectorFieldPComponent)([Real](#) xx, [Real](#) yy))
Binds a given component of a vector field to the grid.
- void [BindVectorFieldQComponent](#) ([Real](#)(*VectorFieldQComponent)([Real](#) xx, [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 Δx .
- [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 Δy .

Friends

- [std::ostream & operator<<](#) ([std::ostream &stream](#), [UniStgGrid2D &in](#))
Prints the grid as a tuple of arrays.

16.19.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 79 of file [mtk_uni_stg_grid_2d.h](#).

16.19.2 Constructor & Destructor Documentation

16.19.2.1 mtk::UniStgGrid2D::UniStgGrid2D ()

Definition at line 131 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.2.2 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](#).

16.19.2.3 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

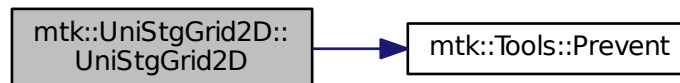
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</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:



16.19.2.4 mtk::UniStgGrid2D::~~UniStgGrid2D ()

Definition at line 203 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.3 Member Function Documentation

16.19.3.1 void mtk::UniStgGrid2D::BindScalarField (Real(*) (Real xx, 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 250 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



16.19.3.2 void mtk::UniStgGrid2D::BindVectorField (Real(*) (Real xx, Real yy) *VectorFieldPComponent*, Real(*) (Real xx, Real yy) *VectorFieldQComponent*)

We assume the field to be of the form:

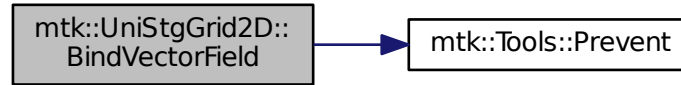
$$\mathbf{v}(x) = p(x,y)\hat{\mathbf{i}} + q(x,y)\hat{\mathbf{j}}$$

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 393 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



16.19.3.3 void mtk::UniStgGrid2D::BindVectorFieldPComponent (Real(*) (Real xx, Real yy) VectorFieldPComponent)
[private]

We assume the field to be of the form:

$$\mathbf{v}(x) = p(x,y)\hat{\mathbf{i}} + q(x,y)\hat{\mathbf{j}}$$

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 300 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.3.4 void mtk::UniStgGrid2D::BindVectorFieldQComponent (Real(*) (Real xx, Real yy) VectorFieldQComponent)
[private]

We assume the field to be of the form:

$$\mathbf{v}(x) = p(x,y)\hat{\mathbf{i}} + q(x,y)\hat{\mathbf{j}}$$

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 365 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.3.5 **mtk::Real** mtk::UniStgGrid2D::delta_x () const

Returns

Computed Δx .

Definition at line 225 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.3.6 **mtk::Real** mtk::UniStgGrid2D::delta_y () const

Returns

Computed Δy .

Definition at line 245 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.3.7 **Real*** mtk::UniStgGrid2D::discrete_domain_x ()

Returns

Pointer to the spatial data.

16.19.3.8 **Real*** mtk::UniStgGrid2D::discrete_domain_y ()

Returns

Pointer to the spatial data.

16.19.3.9 **Real*** mtk::UniStgGrid2D::discrete_field ()

Returns

Pointer to the field data.

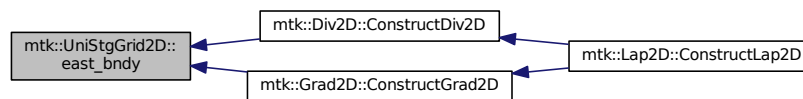
16.19.3.10 **mtk::Real** mtk::UniStgGrid2D::east_bndy () const

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:



16.19.3.11 `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](#).

16.19.3.12 `mtk::Real mtk::UniStgGrid2D::north_bndy () const`

Returns

North boundary spatial coordinate.

Definition at line 235 of file [mtk_uni_stg_grid_2d.cc](#).

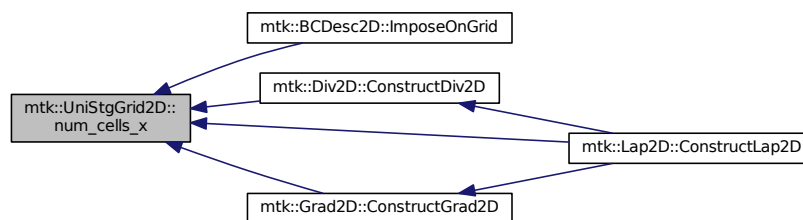
16.19.3.13 `int mtk::UniStgGrid2D::num_cells_x () const`

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:



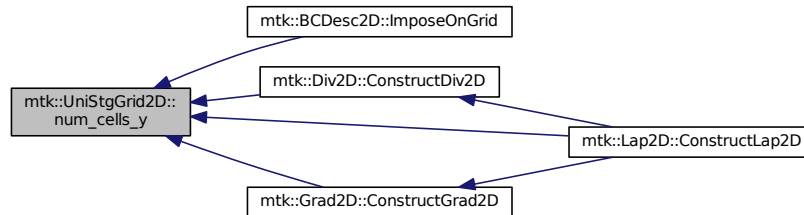
16.19.3.14 `int mtk::UniStgGrid2D::num_cells_y () const`

Returns

Number of cells of the grid.

Definition at line 240 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



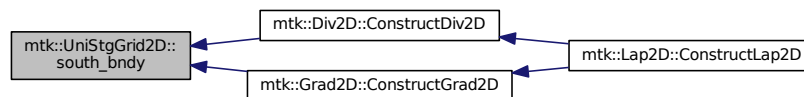
16.19.3.15 mtk::Real mtk::UniStgGrid2D::south_bndy () const

Returns

South boundary spatial coordinate.

Definition at line 230 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



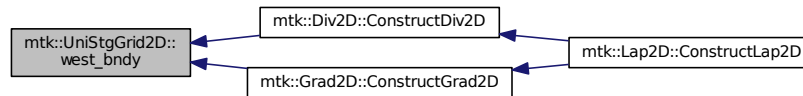
16.19.3.16 mtk::Real mtk::UniStgGrid2D::west_bndy () const

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:



16.19.3.17 `bool mtk::UniStgGrid2D::WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string field_name)`

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 405 of file [mtk_uni_stg_grid_2d.cc](#).

16.19.4 Friends And Related Function Documentation

16.19.4.1 `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](#).

16.19.5 Member Data Documentation

16.19.5.1 **Real** mtk::UniStgGrid2D::delta_x_ [private]

Definition at line 286 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.2 **Real** mtk::UniStgGrid2D::delta_y_ [private]

Definition at line 291 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.3 **std::vector<Real>** mtk::UniStgGrid2D::discrete_domain_x_ [private]

Definition at line 277 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.4 **std::vector<Real>** mtk::UniStgGrid2D::discrete_domain_y_ [private]

Definition at line 278 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.5 **std::vector<Real>** mtk::UniStgGrid2D::discrete_field_ [private]

Definition at line 279 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.6 **Real** mtk::UniStgGrid2D::east_bndy_ [private]

Definition at line 284 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.7 **FieldNature** mtk::UniStgGrid2D::nature_ [private]

Definition at line 281 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.8 **Real** mtk::UniStgGrid2D::north_bndy_ [private]

Definition at line 289 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.9 **int** mtk::UniStgGrid2D::num_cells_x_ [private]

Definition at line 285 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.10 **int** mtk::UniStgGrid2D::num_cells_y_ [private]

Definition at line 290 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.11 **Real** mtk::UniStgGrid2D::south_bndy_ [private]

Definition at line 288 of file [mtk_uni_stg_grid_2d.h](#).

16.19.5.12 **Real** `mtk::UniStgGrid2D::west_bndy_` `[private]`

Definition at line [283](#) 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](#)

Chapter 17

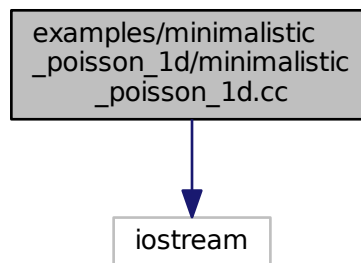
File Documentation

17.1 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` ()

17.1.1 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 parameter.

We consider Robin's boundary conditions of the form:

$$\alpha p(a) - \beta p'(a) = \omega,$$

$$\alpha p(b) + \beta p'(b) = \varepsilon.$$

The analytical solution for this problem is given by

$$p(x) = \frac{\exp(\lambda x) - 1}{\exp(\lambda) - 1}.$$

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

: Raul Vargas-Navarro - vargasna at rohan dot sdsu dot edu

Definition in file [minimalistic_poisson_1d.cc](#).

17.1.2 Function Documentation

17.1.2.1 int main ()

Definition at line 167 of file [minimalistic_poisson_1d.cc](#).

17.2 minimalistic_poisson_1d.cc

```

00001
00042 /*
00043 Copyright (C) 2015, Computational Science Research Center, San Diego State
00044 University. All rights reserved.
00045
00046 Redistribution and use in source and binary forms, with or without modification,
00047 are permitted provided that the following conditions are met:
00048
00049 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00050 and a copy of the modified files should be reported once modifications are
00051 completed, unless these modifications are made through the project's GitHub
00052 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00053 should be developed and included in any deliverable.
00054
00055 2. Redistributions of source code must be done through direct
00056 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00057
00058 3. Redistributions in binary form must reproduce the above copyright notice,
00059 this list of conditions and the following disclaimer in the documentation and/or
00060 other materials provided with the distribution.
00061
00062 4. Usage of the binary form on proprietary applications shall require explicit
00063 prior written permission from the the copyright holders, and due credit should
00064 be given to the copyright holders.
```



```

00065
00066 5. Neither the name of the copyright holder nor the names of its contributors
00067 may be used to endorse or promote products derived from this software without
00068 specific prior written permission.
00069
00070 The copyright holders provide no reassurances that the source code provided does
00071 not infringe any patent, copyright, or any other intellectual property rights of
00072 third parties. The copyright holders disclaim any liability to any recipient for
00073 claims brought against recipient by any third party for infringement of that
00074 parties intellectual property rights.
00075
00076 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00077 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00078 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00079 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00080 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00081 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00082 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00083 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00084 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00085 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00086 */
00087
00088 #if __cplusplus == 201103L
00089
00090 #include <iostream>
00091 #include <fstream>
00092 #include <cmath>
00093 #include <vector>
00094
00095 #include "mtk.h"
00096
00097 mtk::Real Source(mtk::Real xx) {
00098     mtk::Real lambda = -1.0;
00099     return lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00100 }
00101
00102 mtk::Real KnownSolution(mtk::Real xx) {
00103     mtk::Real lambda = -1.0;
00104     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00105 }
00106
00107 int main () {
00108
00109     mtk::Real west_bndy_x = 0.0;
00110     mtk::Real east_bndy_x = 1.0;
00111     mtk::Real relative_norm_2_error{};
00112     int num_cells_x = 5;
00113     mtk::Grad1D grad;
00114     mtk::Lap1D lap;
00115     std::vector<mtk::Real> west_coeffs;
00116     std::vector<mtk::Real> east_coeffs;
00117     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00118     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00119     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00120
00121     if (!lap.ConstructLap1D()) {
00122         std::cerr << "Mimetic lap could not be built." << std::endl;
00123         return EXIT_FAILURE;
00124     }
00125     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00126     if (!grad.ConstructGrad1D()) {
00127         std::cerr << "Mimetic grad could not be built." << std::endl;
00128         return EXIT_FAILURE;
00129     }
00130     mtk::DenseMatrix gradm(grad.ReturnAsDenseMatrix(comp_sol));
00131
00132     source.BindScalarField(Source);
00133
00134     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00135         west_coeffs.push_back(-(exp(-1.0) - 1.0)/-1.0)*gradm.GetValue(0, ii));
00136     }
00137     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00138         east_coeffs.push_back(
00139             (exp(-1.0) - 1.0)/-1.0)*gradm.GetValue(gradm.num_rows() - 1,
00140             gradm.num_cols() - 1 - ii);
00141     }
00142     west_coeffs[0] += -exp(-1.0);
00143     east_coeffs[0] += -exp(-1.0);
00144     mtk::BCDesc1D::ImposeOnOperatorMatrix(lapm, west_coeffs, east_coeffs
00145 );

```

```

00145   mtk::BCDesc1D::ImposeOnGrid(source, -1.0, 0.0);
00146
00147   int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00148   if (info != 0) {
00149       std::cerr << "Something wrong solving system! info = " << info << std::endl;
00150       return EXIT_FAILURE;
00151   }
00152
00153   source.WriteToFile("minimalistic_poisson_1d_comp_sol.dat", "x", "~u(x)");
00154   known_sol.BindScalarField(KnownSolution);
00155   relative_norm_2_error =
00156       mtk::BLASAdapter::RelNorm2Error(source.discrete_field_u(),
00157                                       known_sol.discrete_field_u(),
00158                                       known_sol.num_cells_x());
00159   std::cout << "relative_norm_2_error = ";
00160   std::cout << relative_norm_2_error << std::endl;
00161 }
00162
00163 #else
00164 #include <iostream>
00165 using std::cout;
00166 using std::endl;
00167 int main () {
00168     cout << "This code HAS to be compiled with support for C++11." << endl;
00169     cout << "Exiting..." << endl;
00170     return EXIT_SUCCESS;
00171 }
00172 #endif

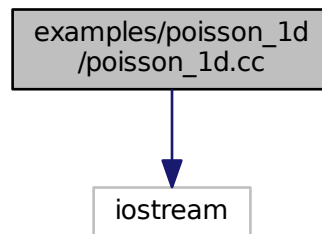
```

17.3 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](#) ()

17.3.1 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 parameter.

We consider Robin's boundary conditions of the form:

$$\alpha p(a) - \beta p'(a) = \omega,$$

$$\alpha p(b) + \beta p'(b) = \varepsilon.$$

The analytical solution for this problem is given by

$$p(x) = \frac{\exp(\lambda x) - 1}{\exp(\lambda) - 1}.$$

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu
 : Raul Vargas-Navarro - vargasna at rohan dot sdsu dot edu

Definition in file [poisson_1d.cc](#).

17.3.2 Function Documentation

17.3.2.1 int main ()

Definition at line 261 of file [poisson_1d.cc](#).

17.4 poisson_1d.cc

```
00001
00042 /*
00043 Copyright (C) 2015, Computational Science Research Center, San Diego State
00044 University. All rights reserved.
00045
00046 Redistribution and use in source and binary forms, with or without modification,
00047 are permitted provided that the following conditions are met:
00048
00049 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00050 and a copy of the modified files should be reported once modifications are
00051 completed, unless these modifications are made through the project's GitHub
00052 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00053 should be developed and included in any deliverable.
00054
00055 2. Redistributions of source code must be done through direct
00056 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00057
00058 3. Redistributions in binary form must reproduce the above copyright notice,
00059 this list of conditions and the following disclaimer in the documentation and/or
00060 other materials provided with the distribution.
00061
00062 4. Usage of the binary form on proprietary applications shall require explicit
```

```

00063 prior written permission from the the copyright holders, and due credit should
00064 be given to the copyright holders.
00065
00066 5. Neither the name of the copyright holder nor the names of its contributors
00067 may be used to endorse or promote products derived from this software without
00068 specific prior written permission.
00069
00070 The copyright holders provide no reassurances that the source code provided does
00071 not infringe any patent, copyright, or any other intellectual property rights of
00072 third parties. The copyright holders disclaim any liability to any recipient for
00073 claims brought against recipient by any third party for infringement of that
00074 parties intellectual property rights.
00075
00076 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00077 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00078 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00079 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00080 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00081 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00082 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00083 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00084 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00085 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00086 */
00087
00088 #if __cplusplus == 201103L
00089
00090 #include <iostream>
00091 #include <fstream>
00092 #include <cmath>
00093
00094 #include <vector>
00095
00096 #include "mtk.h"
00097
00098 mtk::Real Source(mtk::Real xx) {
00099
00100     mtk::Real lambda = -1.0;
00101
00102     return lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00103 }
00104
00105 mtk::Real KnownSolution(mtk::Real xx) {
00106
00107     mtk::Real lambda = -1.0;
00108
00109     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00110 }
00111
00112 int main () {
00113
00114     std::cout << "Example: Poisson Equation on a 1D Uniform Staggered Grid ";
00115     std::cout << "with Robin BCs." << std::endl;
00116
00117     mtk::Real lambda = -1.0;
00118     mtk::Real alpha = -exp(lambda);
00119     mtk::Real beta = (exp(lambda) - 1.0)/lambda;
00120     mtk::Real omega = -1.0;
00121     mtk::Real epsilon = 0.0;
00122
00123     mtk::Real west_bndy_x = 0.0;
00124     mtk::Real east_bndy_x = 1.0;
00125     int num_cells_x = 5;
00126
00127     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00128
00129     int order_of_accuracy{2}; // Desired order of accuracy for approximation.
00130
00131     mtk::Grad1D grad; // Mimetic gradient operator.
00132
00133     mtk::Lapl1D lap; // Mimetic Laplacian operator.
00134
00135     if (!lap.ConstructLapl1D(order_of_accuracy)) {
00136         std::cerr << "Mimetic lap could not be built." << std::endl;
00137         return EXIT_FAILURE;
00138     }
00139
00140     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));

```

```

00147
00148     std::cout << "Mimetic Laplacian operator: " << std::endl;
00149     std::cout << lapm << std::endl;
00150
00151     if (!grad.ConstructGrad1D(order_of_accuracy)) {
00152         std::cerr << "Mimetic grad could not be built." << std::endl;
00153         return EXIT_FAILURE;
00154     }
00155
00156     mtk::DenseMatrix gradm(grad.ReturnAsDenseMatrix(comp_sol));
00157
00158     std::cout << "Mimetic gradient operator: " << std::endl;
00159     std::cout << gradm << std::endl;
00160
00161
00162
00163     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00164
00165     source.BindScalarField(Source);
00166
00167     std::cout << source << std::endl;
00168
00169
00170     // Since we need to approximate the first derivative times beta, we must use
00171     // the approximation of the gradient at the boundary. We could extract them
00172     // from the gradient operator as packed in the grad object. BUT, since we have
00173     // generated at matrix containing this operator, we can extract these from the
00174     // matrix.
00175
00176     // Array containing the coefficients for the west boundary condition.
00177     std::vector<mtk::Real> west_coeffs;
00178
00179     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00180         west_coeffs.push_back(-beta*gradm.GetValue(0, ii));
00181     }
00182
00183     // Array containing the coefficients for the east boundary condition.
00184     std::vector<mtk::Real> east_coeffs;
00185
00186     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00187         east_coeffs.push_back(beta*gradm.GetValue(gradm.num_rows() - 1,
00188                                                    gradm.num_cols() - 1 - ii));
00189     }
00190
00191     // To impose the Dirichlet condition, we simple add its coefficient to the
00192     // first entry of the west, and the last entry of the east array.
00193
00194     west_coeffs[0] += alpha;
00195
00196     east_coeffs[0] += alpha;
00197
00198     // Now that we have the coefficients that should be in the operator, we create
00199     // a boundary condition descriptor object, which will encapsulate the
00200     // complexity of assigning them in the matrix, to complete the construction of
00201     // the mimetic operator.
00202
00203
00204     mtk::BCDesc1D::ImposeOnOperatorMatrix(lapm, west_coeffs, east_coeffs
00205 );
00206
00207     std::cout << "Mimetic Laplacian with Robin conditions:" << std::endl;
00208     std::cout << lapm << std::endl;
00209
00210     mtk::BCDesc1D::ImposeOnGrid(source, omega, epsilon);
00211
00212     std::cout << "Source term with imposed BCs:" << std::endl;
00213     std::cout << source << std::endl;
00214
00215     source.WriteToFile("poisson_1d_source.dat", "x", "s(x)");
00216
00217
00218     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00219
00220     if (!info) {
00221         std::cout << "System solved! Problem solved!" << std::endl;
00222         std::cout << std::endl;
00223     }
00224     else {
00225         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00226         std::cerr << "Exiting..." << std::endl;
00227         return EXIT_FAILURE;
00228     }
00229

```

```

00230     std::cout << "Computed solution:" << std::endl;
00231     std::cout << source << std::endl;
00232
00233     source.WriteToFile("poisson_1d_comp_sol.dat", "x", "~u(x)");
00234
00235
00236
00237     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00238
00239     known_sol.BindScalarField(KnownSolution);
00240
00241     std::cout << "known_sol =" << std::endl;
00242     std::cout << known_sol << std::endl;
00243
00244     known_sol.WriteToFile("poisson_1d_known_sol.dat", "x", "u(x)");
00245
00246     mtk::Real relative_norm_2_error{}; // Relative norm 2 of the error.
00247
00248     relative_norm_2_error =
00249         mtk::BLASAdapter::RelNorm2Error(source.discrete_field_u(),
00250                                         known_sol.discrete_field_u(),
00251                                         known_sol.num_cells_x());
00252
00253     std::cout << "relative_norm_2_error = ";
00254     std::cout << relative_norm_2_error << std::endl;
00255 }
00256
00257 #else
00258 #include <iostream>
00259 using std::cout;
00260 using std::endl;
00261 int main () {
00262     cout << "This code HAS to be compiled with support for C++11." << endl;
00263     cout << "Exiting..." << endl;
00264     return EXIT_SUCCESS;
00265 }
00266 #endif

```

17.5 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_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
#include "mtk_interp_1d.h"
#include "mtk_quad_1d.h"
#include "mtk_bc_desc_1d.h"
#include "mtk_grad_2d.h"

```



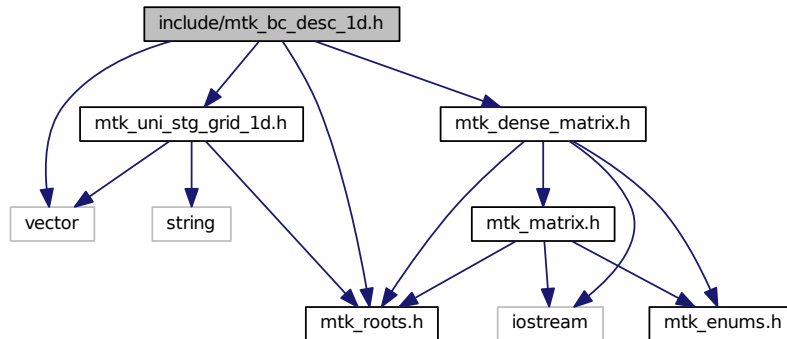
```
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
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 */
00379 #ifndef MTK_INCLUDE_MTK_H_
00380 #define MTK_INCLUDE_MTK_H_
00381
00389 #include "mtk_roots.h"
00390
00398 #include "mtk_enums.h"
00399
00407 #include "mtk_tools.h"
00408
00416 #include "mtk_matrix.h"
00417 #include "mtk_dense_matrix.h"
00418
00426 #include "mtk_blas_adapter.h"
00427 #include "mtk_lapack_adapter.h"
00428 #include "mtk_glpk_adapter.h"
00429
00437 #include "mtk_uni_stg_grid_1d.h"
00438 #include "mtk_uni_stg_grid_2d.h"
00439
00447 #include "mtk_grad_1d.h"
00448 #include "mtk_div_1d.h"
00449 #include "mtk_lap_1d.h"
00450 #include "mtk_interp_1d.h"
00451 #include "mtk_quad_1d.h"
00452 #include "mtk_bc_desc_1d.h"
00453
00454 #include "mtk_grad_2d.h"
00455
00456 #endif // End of: MTK_INCLUDE_MTK_H_
```

17.7 include/mtk_bc_desc_1d.h File Reference

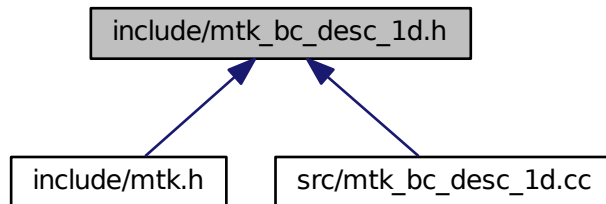
Enforces boundary conditions in either the operator or the grid.

```
#include <vector>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```


Include dependency graph for mtk_bc_desc_1d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::BCDesc1D](#)

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

17.7.1 Detailed Description

This class presents an interface for the user to specify boundary conditions on 1D mimetic operators and the grids they are acting on.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_bc_desc_1d.h](#).

17.8 mtk_bc_desc_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 #include <vector>
00058
00059 #include "mtk_roots.h"
00060 #include "mtk_dense_matrix.h"
00061 #include "mtk_uni_stg_grid_1d.h"
00062
00063 #ifndef MTK_INCLUDE_BC_DESC_1D_H_
00064 #define MTK_INCLUDE_BC_DESC_1D_H_
00065
00066 namespace mtk {
00067
00068 class BCDesc1D {
00069 public:
00070     static void ImposeOnOperatorMatrix(DenseMatrix &matrix,
00071                                         const std::vector<Real> &west,
00072                                         const std::vector<Real> &east);
00073
00074     static void ImposeOnGrid(UniStgGrid1D &grid,
00075                              const Real &omega,
00076                              const Real &epsilon);
00077 };
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091

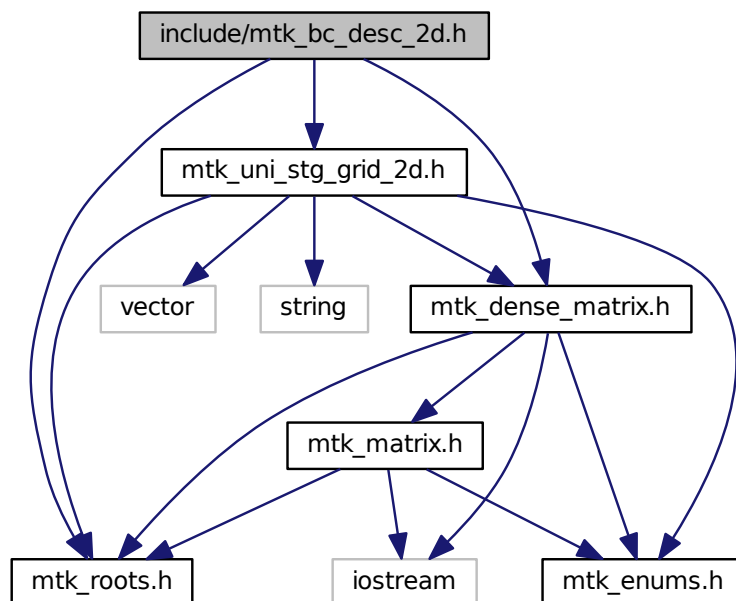
```

```
00092 }  
00093 #endif // End of: MTK_INCLUDE_BC_DESC_1D_H_
```

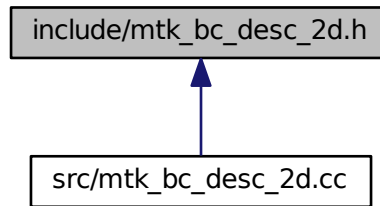
17.9 include/mtk_bc_desc_2d.h File Reference

Enforces boundary conditions in either the operator or the grid.

```
#include "mtk_roots.h"  
#include "mtk_dense_matrix.h"  
#include "mtk_uni_stg_grid_2d.h"  
Include dependency graph for mtk_bc_desc_2d.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::BCDesc2D](#)

Namespaces

- [mtk](#)
Mimetic Methods Toolkit namespace.

17.9.1 Detailed Description

This class presents an interface for the user to specify boundary conditions on 2D mimetic operators and the grids they are acting on.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_bc_desc_2d.h](#).

17.10 mtk_bc_desc_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_BC_DESC_2D_H_
00058 #define MTK_INCLUDE_BC_DESC_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
00066 class BCDesc2D {
00067 public:
00077     static void ImposeOnOperatorMatrix(const DenseMatrix &matrix,
00078                                         Real (*west)(int ii, int jj),
00079                                         Real (*east)(int ii, int jj),
00080                                         Real (*south)(int ii, int jj),
00081                                         Real (*north)(int ii, int jj));
00082
00092     static void ImposeOnGrid(const UniStgGrid2D &grid,
00093                               Real (*west)(Real xx, Real yy),
00094                               Real (*east)(Real xx, Real yy),
00095                               Real (*south)(Real xx, Real yy),
00096                               Real (*north)(Real xx, Real yy));
00097 };
00098
00099 #endif // End of: MTK_INCLUDE_BC_DESC_2D_H_

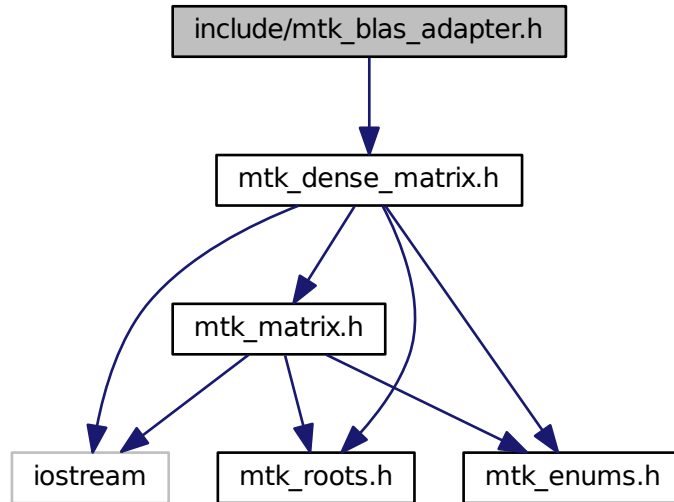
```

17.11 include/mtk_blas_adapter.h File Reference

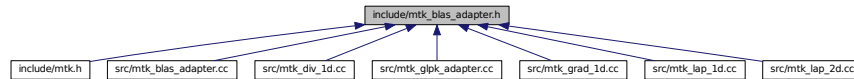
Adapter class for the BLAS API.

```
#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.

17.11.1 Detailed Description

This class contains a collection of static classes, 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](#).

17.12 mtk_blas_adapter.h

```
00001
00024 /*
00025 Copyright (C) 2015, Computational Science Research Center, San Diego State
00026 University. All rights reserved.
00027
00028 Redistribution and use in source and binary forms, with or without modification,
00029 are permitted provided that the following conditions are met:
00030
00031 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00032 and a copy of the modified files should be reported once modifications are
00033 completed, unless these modifications are made through the project's GitHub
00034 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00035 should be developed and included in any deliverable.
00036
00037 2. Redistributions of source code must be done through direct
00038 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00039
00040 3. Redistributions in binary form must reproduce the above copyright notice,
00041 this list of conditions and the following disclaimer in the documentation and/or
00042 other materials provided with the distribution.
00043
00044 4. Usage of the binary form on proprietary applications shall require explicit
00045 prior written permission from the the copyright holders, and due credit should
00046 be given to the copyright holders.
00047
00048 5. Neither the name of the copyright holder nor the names of its contributors
00049 may be used to endorse or promote products derived from this software without
00050 specific prior written permission.
00051
00052 The copyright holders provide no reassurances that the source code provided does
00053 not infringe any patent, copyright, or any other intellectual property rights of
00054 third parties. The copyright holders disclaim any liability to any recipient for
00055 claims brought against recipient by any third party for infringement of that
00056 parties intellectual property rights.
00057
00058 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00059 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00060 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00061 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00062 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00063 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00064 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00065 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00066 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00071 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00072
00073 #include "mtk_dense_matrix.h"
```

```

00074
00075 namespace mtk {
00076
00096 class BLASAdapter {
00097 public:
00106     static Real RealNRM2(Real *in, int &in_length);
00107
00124     static void RealAXPY(Real alpha, Real *xx, Real *yy, int &in_length);
00125
00140     static Real RelNorm2Error(Real *computed, Real *known, int length);
00141
00159     static void RealDenseMV(Real &alpha,
00160                             DenseMatrix &aa,
00161                             Real *xx,
00162                             Real &beta,
00163                             Real *yy);
00164
00179     static DenseMatrix RealDenseMM(DenseMatrix &aa,
00180                                   DenseMatrix &bb);
00181 }
00182 #endif // End of: MTK_INCLUDE_BLAS_ADAPTER_H_

```

17.13 include/mtk_dense_matrix.h File Reference

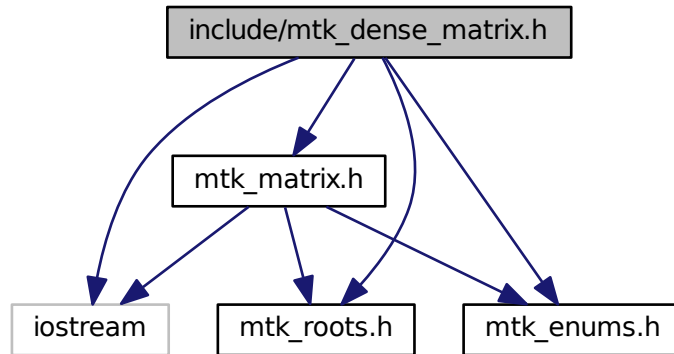
Defines a common dense matrix, using a 1D array.

```

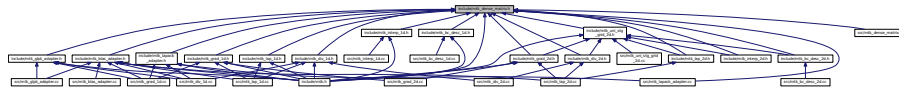
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_matrix.h"

```

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.

17.13.1 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](#).

17.14 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
00098     DenseMatrix& operator =(const DenseMatrix &in);
00099
00101     bool operator ==(const DenseMatrix &in);
00102
00104     DenseMatrix();
00105
00111     DenseMatrix(const DenseMatrix &in);
00112
00121     DenseMatrix(const int &num_rows, const int &num_cols);
00122
00148     DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00149
00183     DenseMatrix(const Real *gen,
00184                 const int &gen_length,
00185                 const int &pro_length,
00186                 const bool &transpose);
00187
00189     ~DenseMatrix();
00190
00196     Matrix matrix_properties() const;
00197
00203     int num_rows() const;
00204
00210     int num_cols() const;
00211
00217     Real* data() const;
00218
00226     void SetOrdering(mtk::MatrixOrdering oo);
00227
00236     Real GetValue(const int &row_coord, const int &col_coord) const;
00237
00245     void SetValue(const int &row_coord,
00246                  const int &col_coord,
00247                  const Real &val);
00248
00250     void Transpose();
00251
00253     void OrderRowMajor();
00254
00256     void OrderColMajor();
00257
00268     static DenseMatrix Kron(const DenseMatrix &aa, const
DenseMatrix &bb);
00269
00279     bool WriteToFile(std::string filename);

```

```

00280
00281 private:
00282     Matrix matrix_properties;
00283
00284     Real *data_;
00285 };
00286 }
00287 #endif // End of: MTK_INCLUDE_MTK_DENSE_MATRIX_H_

```

17.15 include/mtk_div_1d.h File Reference

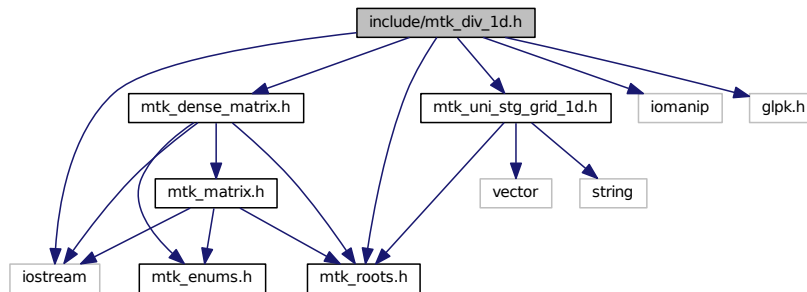
Includes the definition of the class Div1D.

```

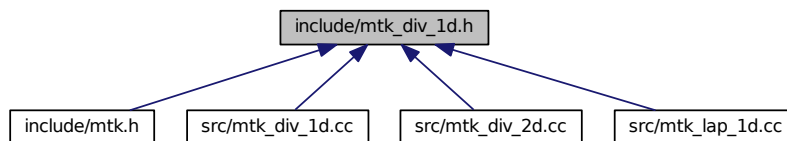
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk_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.

17.15.1 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](#).

17.16 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>

```

```

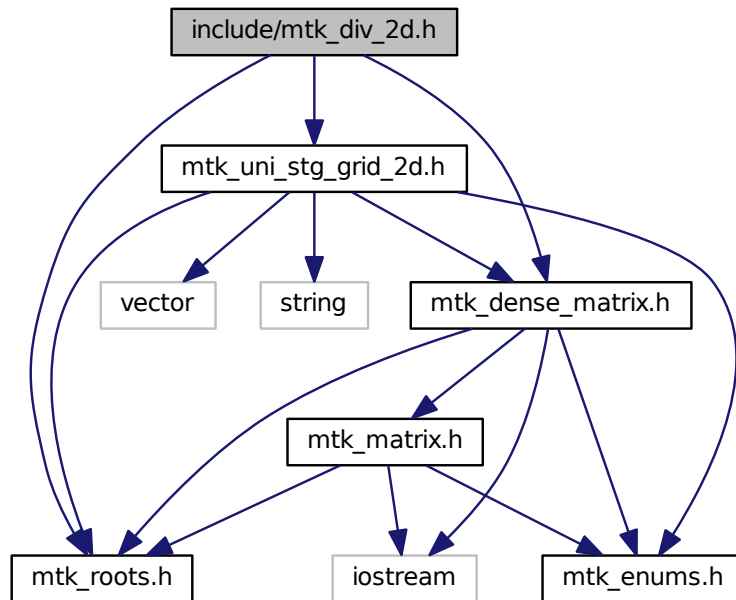
00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Div1D {
00082 public:
00084     friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00085
00087     Div1D();
00088
00094     Div1D(const Div1D &div);
00095
00097     ~Div1D();
00098
00104     bool ConstructDiv1D(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(const
    UniStgGrid1D &grid);
00148
00149 private:
00155     bool ComputeStencilInteriorGrid(void);
00156
00163     bool ComputeRationalBasisNullSpace(void);
00164
00170     bool ComputePreliminaryApproximations(void);
00171
00177     bool ComputeWeights(void);
00178
00184     bool ComputeStencilBoundaryGrid(void);
00185
00191     bool AssembleOperator(void);
00192
00193     int order_accuracy_;
00194     int dim_null_;
00195     int num_bndy_coeffs_;
00196     int divergence_length_;
00197     int minrow_;
00198     int row_;
00199
00200     DenseMatrix rat_basis_null_space_;
00201
00202     Real *coeffs_interior_;
00203     Real *prem_apps_;
00204     Real *weights_crs_;
00205     Real *weights_cbs_;
00206     Real *mim_bndy_;
00207     Real *divergence_;
00208
00209     Real mimetic_threshold_;
00210 };
00211 }
00212 #endif // End of: MTK_INCLUDE_DIV_1D_H_

```

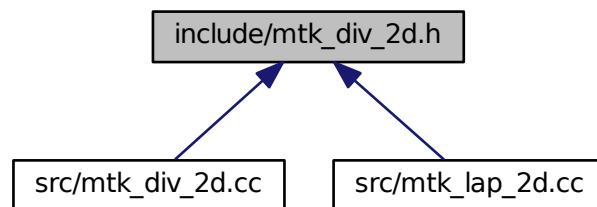
17.17 include/mtk_div_2d.h File Reference

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](#)

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

17.17.1 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](#).

17.18 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
00066 class Div2D {
00067 public:
00068     Div2D();
00069
00070     Div2D(const Div2D &div);
00071
00072     ~Div2D();
00073
00074     bool ConstructDiv2D(const UniStgGrid2D &grid,
00075                        int order_accuracy = kDefaultOrderAccuracy,
00076                        Real mimetic_threshold = kDefaultMimeticThreshold);
00077
00078     DenseMatrix ReturnAsDenseMatrix();
00079 private:
00080     DenseMatrix divergence_;
00081     int order_accuracy_;
00082     Real mimetic_threshold_;
00083 };
00084 }
00085 #endif // End of: MTK_INCLUDE_MTK_DIV_2D_H_

```

17.19 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::DENSE](#), [mtk::BANDED](#), [mtk::CRS](#) }
Considered matrix storage schemes to implement sparse matrices.
- enum [mtk::MatrixOrdering](#) { [mtk::ROW_MAJOR](#), [mtk::COL_MAJOR](#) }
Considered matrix ordering (for Fortran purposes).
- enum [mtk::FieldNature](#) { [mtk::SCALAR](#), [mtk::VECTOR](#) }
Nature of the field discretized in a given grid.
- enum [mtk::DirInterp](#) { [mtk::SCALAR_TO_VECTOR](#), [mtk::VECTOR_TO_SCALAR](#) }
1D interpolation operator.

17.19.1 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](#).

17.20 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
00077 enum MatrixStorage {
00078     DENSE,
00079     BANDED,
00080     CRS
00081 };
00082
00095 enum MatrixOrdering {
00096     ROW_MAJOR,

```

```

00097     COL_MAJOR
00098 };
00099
00113 enum FieldNature {
00114     SCALAR,
00115     VECTOR
00116 };
00117
00127 enum DirInterp {
00128     SCALAR_TO_VECTOR,
00129     VECTOR_TO_SCALAR
00130 };
00131 }
00132 #endif // End of: MTK_INCLUDE_ENUMS_H_

```

17.21 include/mtk_glpk_adapter.h File Reference

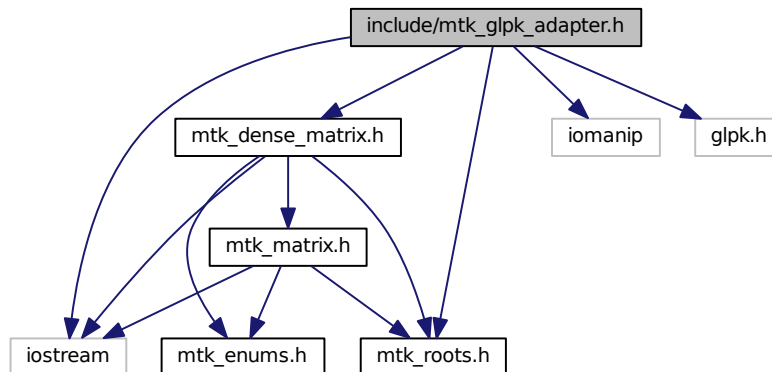
Adapter class for the GLPK API.

```

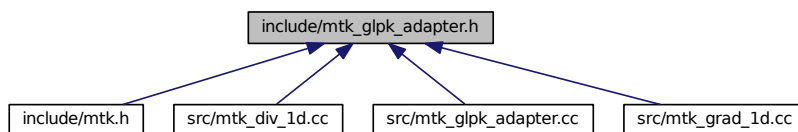
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"

```

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.

17.21.1 Detailed Description

This class contains a collection of static classes, 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`.

17.22 mtk_glpk_adapter.h

```

00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
00026 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00027 and a copy of the modified files should be reported once modifications are
00028 completed, unless these modifications are made through the project's GitHub
00029 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00030 should be developed and included in any deliverable.
00031
00032 2. Redistributions of source code must be done through direct
00033 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00034
00035 3. Redistributions in binary form must reproduce the above copyright notice,
00036 this list of conditions and the following disclaimer in the documentation and/or
00037 other materials provided with the distribution.
00038
00039 4. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders, and due credit should
00041 be given to the copyright holders.
00042
00043 5. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does

```

```

00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00066 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00067
00068 #include <iostream>
00069 #include <iomanip>
00070
00071 #include "glpk.h"
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_dense_matrix.h"
00075
00076 namespace mtk {
00077
00101 class GLPKAdapter {
00102 public:
00123     static mtk::Real SolveSimplexAndCompare(
00124         mtk::Real *A,
00125         int nrows,
00126         int ncols,
00127         int kk,
00128         mtk::Real *hh,
00129         mtk::Real *qq,
00130         int robjective,
00131         mtk::Real mimetic_tol,
00132         int copy);
00133 };
00134 #endif // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_

```

17.23 include/mtk_grad_1d.h File Reference

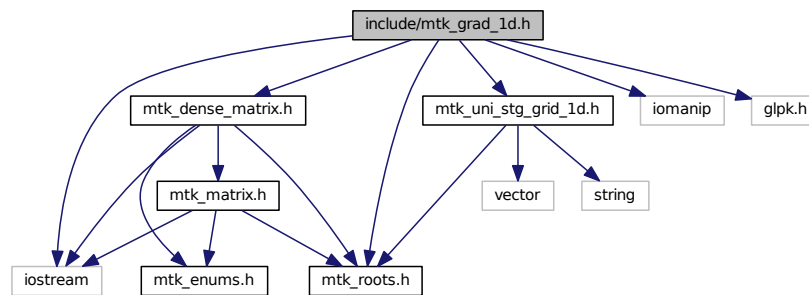
Includes the definition of the class Grad1D.

```

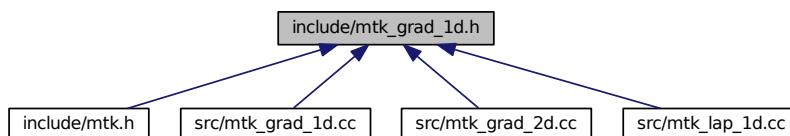
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.

17.23.1 Detailed Description

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

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_1d.h](#).

17.24 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,
Real east, int num_cells_x);
00148
00154     DenseMatrix ReturnAsDenseMatrix(const
UniStgGrid1D &grid);
00155
00161     DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x);
00162
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_

```

17.25 include/mtk_grad_2d.h File Reference

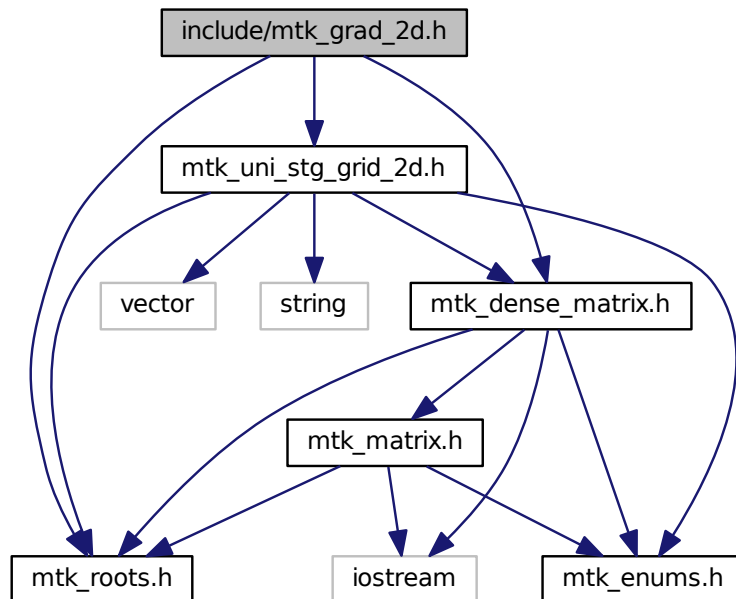
Includes the definition of the class Grad2D.

```

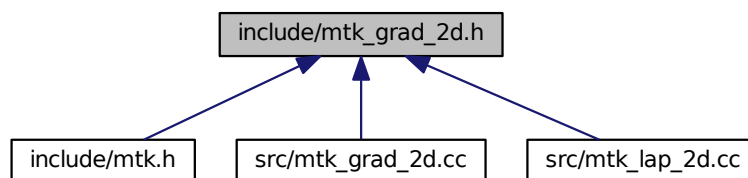
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for `mtk_grad_2d.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::Grad2D](#)

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

17.25.1 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](#).

17.26 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
00066 class Grad2D {
00067 public:
00068     Grad2D();
00070

```

```

00076   Grad2D(const Grad2D &grad);
00077
00078   ~Grad2D();
00079
00080   bool ConstructGrad2D(const UniStgGrid2D &grid,
00081                       int order_accuracy = kDefaultOrderAccuracy,
00082                       Real mimetic_threshold = kDefaultMimeticThreshold);
00083
00084   DenseMatrix ReturnAsDenseMatrix();
00085
00086 private:
00087   DenseMatrix gradient_;
00088   int order_accuracy_;
00089   Real mimetic_threshold_;
00090 };
00091
00092 #endif // End of: MTK_INCLUDE_MTK_GRAD_2D_H_

```

17.27 include/mtk_interp_1d.h File Reference

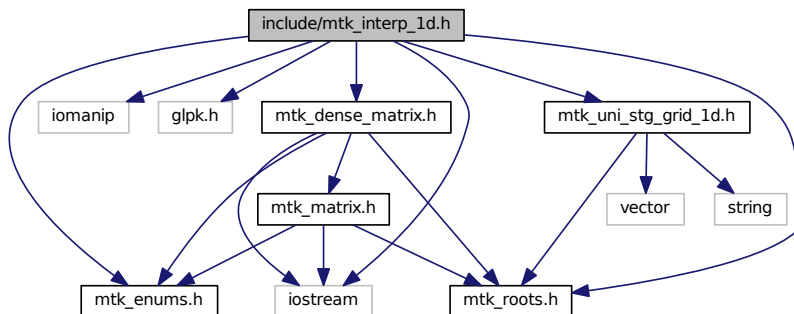
Includes the definition of the class Interp1D.

```

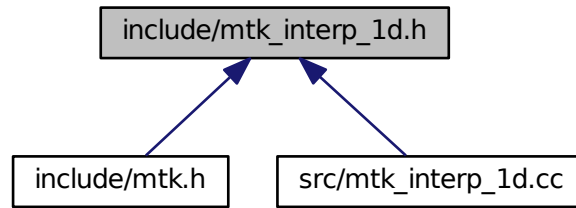
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.

17.27.1 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`.

17.28 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

```

```

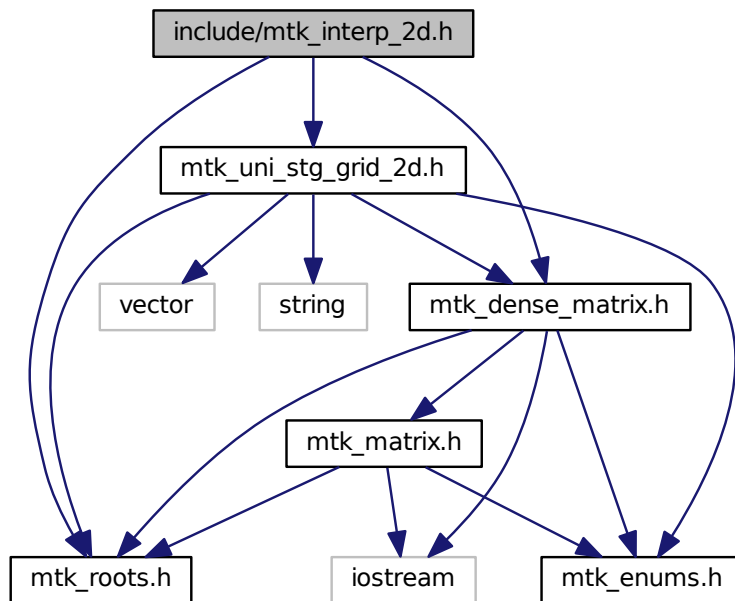
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.cs.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the 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 #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);
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_

```

17.29 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](#)

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

17.29.1 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](#).

17.30 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
00067 class Interp2D {
00068 public:
00069     Interp2D();
00070
00071     Interp2D(const Interp2D &interp);
00072
00073     ~Interp2D();
00074
00075     DenseMatrix ConstructInterp2D(const UniStgGrid2D &grid,
00076                                 int order_accuracy = kDefaultOrderAccuracy,

```

```

00089             Real mimetic_threshold =
00090             kDefaultMimeticThreshold);
00096     DenseMatrix ReturnAsDenseMatrix();
00097
00098     private:
00099     DenseMatrix interpolator_;
00100
00101     int order_accuracy_;
00102
00103     Real mimetic_threshold_;
00104 };
00105 }
00106 #endif // End of: MTK_INCLUDE_MTK_INTERP_2D_H_

```

17.31 include/mtk_lap_1d.h File Reference

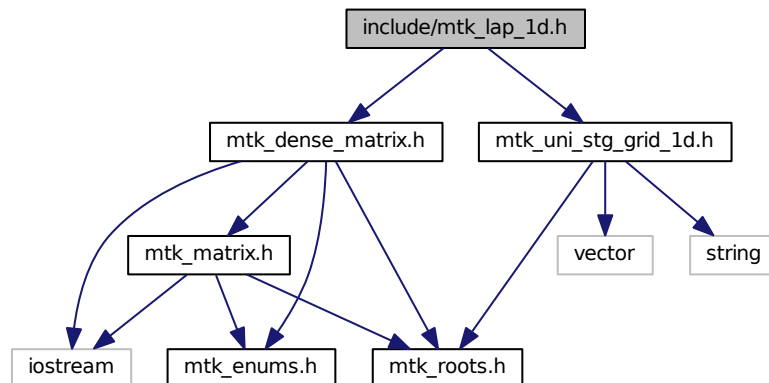
Includes the definition of the class Lap1D.

```

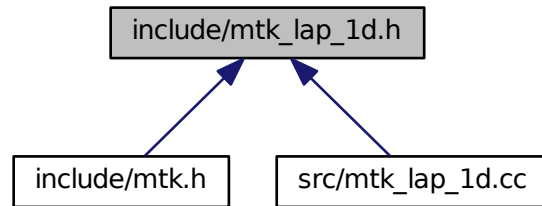
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.

17.31.1 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`.

17.32 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

```



```

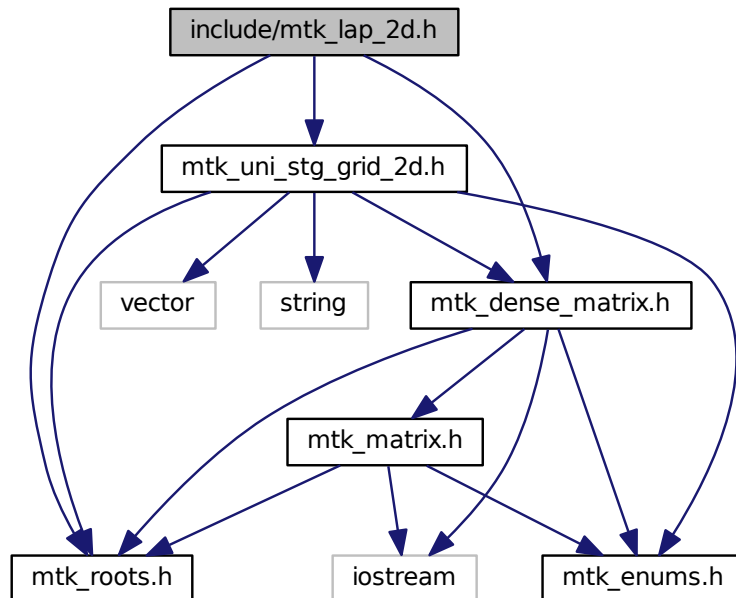
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation 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
00076 class Lap1D {
00077 public:
00078     friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00079
00080     Lap1D();
00081
00082     Lap1D(const Lap1D &lap);
00083
00084     ~Lap1D();
00085
00086     bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00087                        Real mimetic_threshold = kDefaultMimeticThreshold);
00088
00089     DenseMatrix ReturnAsDenseMatrix(const
00090     UniStgGrid1D &grid);
00091
00092     mtk::Real* data(const UniStgGrid1D &grid);
00093
00094 private:
00095     int order_accuracy_;
00096     int laplacian_length_;
00097     Real *laplacian_;
00098     Real mimetic_threshold_;
00099 };
00100
00101 #endif // End of: MTK_INCLUDE_LAP_1D_H_

```

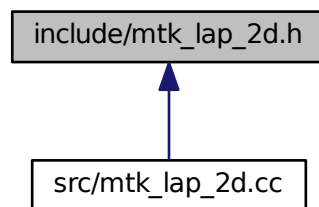
17.33 include/mtk_lap_2d.h File Reference

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](#)

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

17.33.1 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](#).

17.34 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
00066 class Lap2D {
00067 public:
00068     Lap2D();
00069
00070     Lap2D(const Lap2D &lap);
00071
00072     ~Lap2D();
00073
00074     bool ConstructLap2D(const UniStgGrid2D &grid,
00075                        int order_accuracy = kDefaultOrderAccuracy,
00076                        Real mimetic_threshold = kDefaultMimeticThreshold);
00077
00078     DenseMatrix ReturnAsDenseMatrix();
00079
00080     mtk::Real* data();
00081
00082 private:
00083     DenseMatrix laplacian_;
00084     int order_accuracy_;
00085     Real mimetic_threshold_;
00086 };
00087
00088 #endif // End of: MTK_INCLUDE_MTK_LAP_2D_H_

```

17.35 include/mtk_lapack_adapter.h File Reference

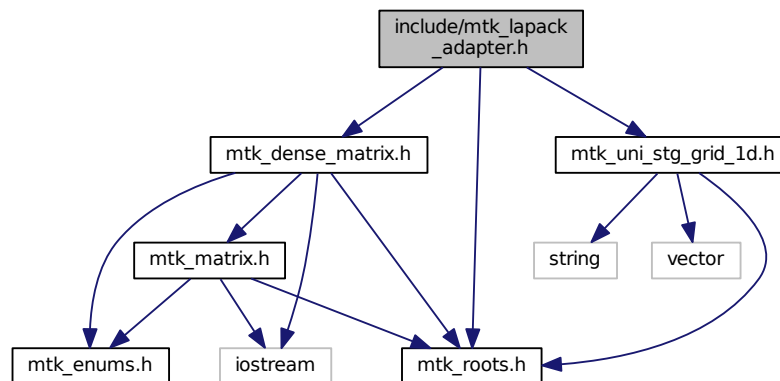
Adapter class for the LAPACK API.

```

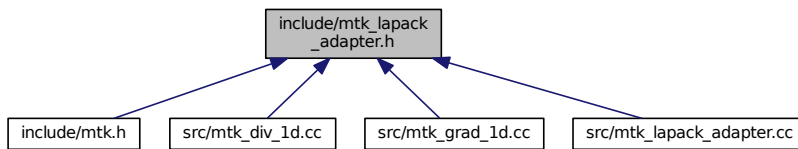
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include 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.

17.35.1 Detailed Description

This class contains a collection of static classes, 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](#).

17.36 mtk_lapack_adapter.h

```

00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
00026 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00027 and a copy of the modified files should be reported once modifications are

```

```

00028 completed, unless these modifications are made through the project's GitHub
00029 page: http://www.cs.csrc.sdsu.edu/mtk. Documentation related to said modifications
00030 should be developed and included in any deliverable.
00031
00032 2. Redistributions of source code must be done through direct
00033 downloads from the project's GitHub page: http://www.cs.csrc.sdsu.edu/mtk
00034
00035 3. Redistributions in binary form must reproduce the above copyright notice,
00036 this list of conditions and the following disclaimer in the documentation and/or
00037 other materials provided with the distribution.
00038
00039 4. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders, and due credit should
00041 be given to the copyright holders.
00042
00043 5. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does
00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00066 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00067
00068 #include "mtk_roots.h"
00069 #include "mtk_dense_matrix.h"
00070 #include "mtk_uni_stg_grid_ld.h"
00071
00072 namespace mtk {
00073
00092 class LAPACKAdapter {
00093 public:
00104     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00105                                mtk::Real *rhs);
00106
00117     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00118                                mtk::DenseMatrix &rr);
00119
00130     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00131                                mtk::UniStgGridLD &rhs);
00132
00144     static int SolveRectangularDenseSystem(const
00145 mtk::DenseMatrix &aa,
00146                                           mtk::Real *ob_,
00147                                           int ob_ld_);
00159     static mtk::DenseMatrix QRFactorDenseMatrix(
00160 DenseMatrix &matrix);
00161 };
00162 #endif // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_

```

17.37 include/mtk_matrix.h File Reference

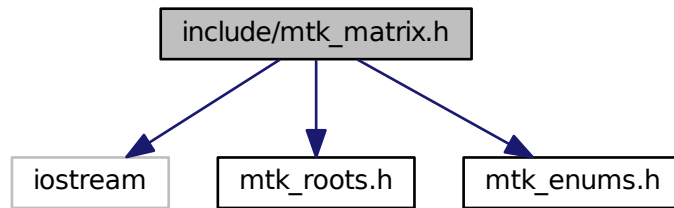
Definition of the representation of a matrix in the MTK.

```

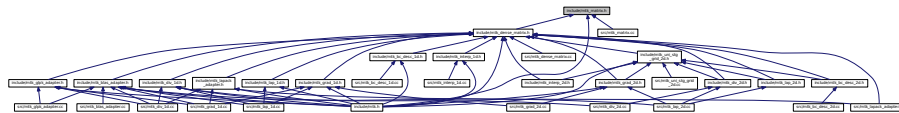
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"

```

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.

17.37.1 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](#).

17.38 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();
00082
00083     MatrixStorage storage() const;
00084
00085     MatrixOrdering ordering() const;
00086
00087     int num_rows() const;
00088
00089     int num_cols() const;
00090
00091     int num_values() const;
00092
00093     int ld() const;
00094
00095     int num_zero() const;
00096
00097     int num_non_zero() const;
00098
00099     int num_null() const;
00100
00101     int num_non_null() const;

```



```

00167
00173     int kl() const;
00174
00180     int ku() const;
00181
00187     int bandwidth() const;
00188
00196     Real abs_density() const;
00197
00205     Real rel_density() const;
00206
00214     Real abs_sparsity() const;
00215
00223     Real rel_sparsity() const;
00224
00232     void set_storage(const MatrixStorage &tt);
00233
00241     void set_ordering(const MatrixOrdering &oo);
00242
00248     void set_num_rows(int num_rows);
00249
00255     void set_num_cols(int num_cols);
00256
00262     void set_num_zero(int in);
00263
00269     void set_num_null(int in);
00270
00272     void IncreaseNumZero();
00273
00275     void IncreaseNumNull();
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_

```

17.39 include/mtk_quad_1d.h File Reference

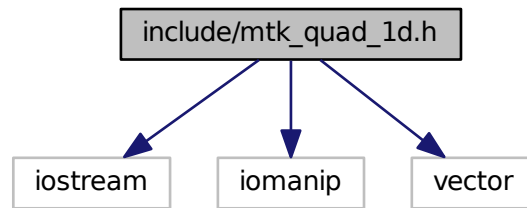
Includes the definition of the class Quad1D.

```

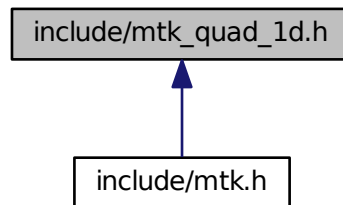
#include <iostream>
#include <iomanip>
#include <vector>

```

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.

17.39.1 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](#).

17.40 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);
00122
00123 private:
00124 int degree_approximation_;
00125
00126 std::vector<Real> weights_;
00127 };
00128 }
00129 #endif // End of: MTK_INCLUDE_QUAD_1D_H_

```

17.41 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::kDefaultTolerance](#) {1e-7f}
Considered tolerance for comparisons in numerical methods.
- const int [mtk::kDefaultOrderAccuracy](#) {2}
Default order of accuracy for mimetic operators.
- const float [mtk::kDefaultMimeticThreshold](#) {1e-6f}
Default tolerance for higher-order mimetic operators.
- const int [mtk::kCriticalOrderAccuracyDiv](#) {8}
At this order (and higher) we must use the CBSA to construct.
- const int [mtk::kCriticalOrderAccuracyGrad](#) {10}
At this order (and higher) we must use the CBSA to construct.

17.41.1 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 Documentation should (better?) capture effects from selective compilation.

Todo Test selective precision mechanisms.

Definition in file [mtk_roots.h](#).

17.42 mtk_roots.h

```

00001
00017 /*
00018 Copyright (C) 2015, Computational Science Research Center, San Diego State
00019 University. All rights reserved.
00020
00021 Redistribution and use in source and binary forms, with or without modification,
00022 are permitted provided that the following conditions are met:
00023
00024 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00025 and a copy of the modified files should be reported once modifications are
00026 completed, unless these modifications are made through the project's GitHub
00027 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00028 should be developed and included in any deliverable.
00029
00030 2. Redistributions of source code must be done through direct
00031 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00032
00033 3. Redistributions in binary form must reproduce the above copyright notice,
00034 this list of conditions and the following disclaimer in the documentation and/or
00035 other materials provided with the distribution.
00036
00037 4. Usage of the binary form on proprietary applications shall require explicit
00038 prior written permission from the the copyright holders, and due credit should
00039 be given to the copyright holders.
00040
00041 5. Neither the name of the copyright holder nor the names of its contributors
00042 may be used to endorse or promote products derived from this software without
00043 specific prior written permission.
00044
00045 The copyright holders provide no reassurances that the source code provided does
00046 not infringe any patent, copyright, or any other intellectual property rights of
00047 third parties. The copyright holders disclaim any liability to any recipient for
00048 claims brought against recipient by any third party for infringement of that
00049 parties intellectual property rights.
00050
00051 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00052 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00053 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00054 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00055 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00056 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00057 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00058 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00059 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00060 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00061 */
00062
00063 #ifndef MTK_INCLUDE_ROOTS_H_
00064 #define MTK_INCLUDE_ROOTS_H_
00065
00071 namespace mtk {

```

```

00072
00080 #ifndef MTK_PRECISION_DOUBLE
00081 typedef double Real;
00082 #else
00083 typedef float Real;
00084 #endif
00085
00103 #ifndef MTK_PRECISION_DOUBLE
00104 const double kZero{0.0};
00105 const double kOne{1.0};
00106 #else
00107 const float kZero{0.0f};
00108 const float kOne{1.0f};
00109 #endif
00110
00118 #ifndef MTK_PRECISION_DOUBLE
00119 const double kDefaultTolerance{1e-7};
00120 #else
00121 const float kDefaultTolerance{1e-7f};
00122 #endif
00123
00133 const int kDefaultOrderAccuracy{2};
00134
00144 #ifndef MTK_PRECISION_DOUBLE
00145 const double kDefaultMimeticThreshold{1e-6};
00146 #else
00147 const float kDefaultMimeticThreshold{1e-6f};
00148 #endif
00149
00157 const int kCriticalOrderAccuracyDiv{8};
00158
00166 const int kCriticalOrderAccuracyGrad{10};
00167 }
00168 #endif // End of: MTK_INCLUDE_ROOTS_H_

```

17.43 include/mtk_tools.h File Reference

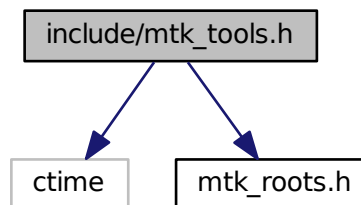
Tool manager class.

```

#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.

17.43.1 Detailed Description

Basic utilities.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file `mtk_tools.h`.

17.44 mtk_tools.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_TOOLS_H_
00057 #define MTK_INCLUDE_TOOLS_H_
00058
00059 #include <ctime>
00060
00061 #include "mtk_roots.h"
00062
00063 namespace mtk {
00064
00074 class Tools {
00075 public:
00086     static void Prevent(const bool complement,
00087                         const char *fname,
00088                         int lineno,
00089                         const char *fxname);
00090
00096     static void BeginUnitTestNo(const int &nn);
00097
00103     static void EndUnitTestNo(const int &nn);
00104
00110     static void Assert(const bool condition);
00111 private:
00112     static int test_number_;
00114     static Real duration_;
00115     static clock_t begin_time_;
00118 };
00119 }
00120 #endif // End of: MTK_INCLUDE_TOOLS_H_

```

17.45 include/mtk_uni_stg_grid_1d.h File Reference

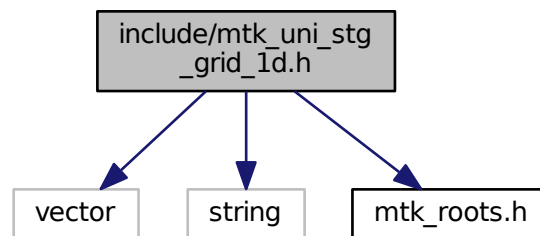
Definition of an 1D uniform staggered grid.

```

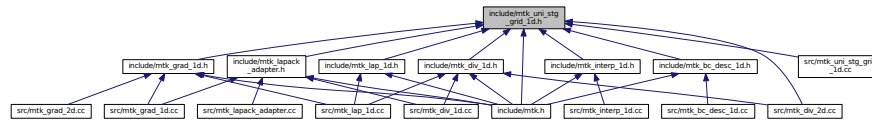
#include <vector>
#include <string>
#include "mtk_roots.h"

```

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.

17.45.1 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`.

17.46 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 namespace mtk {
00066
00067 class UniStgGrid1D {
00068 public:
00080     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
00082     UniStgGrid1D();
00083
00084     UniStgGrid1D(const UniStgGrid1D &grid);
00085
00086     UniStgGrid1D(const Real &west_bndy_x,
00087                  const Real &east_bndy_x,
00088                  const int &num_cells_x,
00089                  const mtk::FieldNature &nature = mtk::SCALAR);
00090
00091     ~UniStgGrid1D();
00092
00093     Real west_bndy_x() const;
00094
00095     Real east_bndy_x() const;
00096
00097     Real delta_x() const;
00098
00099     Real *discrete_domain_x();
00100
00101     Real *discrete_field_u();
00102
00103     int num_cells_x() const;
00104
00105     void BindScalarField(Real (*ScalarField)(Real xx));
00106
00107     void BindVectorField(Real (*VectorField)(Real xx));
00108
00109     bool WriteToFile(std::string filename,
00110                     std::string space_name,
00111                     std::string field_name);
00112
00113 private:
00114     FieldNature nature_;
00115
00116     std::vector<Real> discrete_domain_x_;
00117     std::vector<Real> discrete_field_u_;
00118
00119     Real west_bndy_x_;
00120     Real east_bndy_x_;
00121     Real num_cells_x_;
00122     Real delta_x_;
00123 };
00124
00125 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_

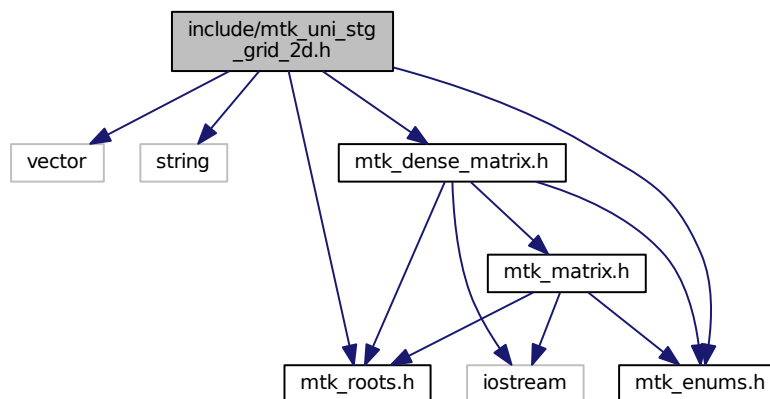
```

17.47 include/mtk_uni_stg_grid_2d.h File Reference

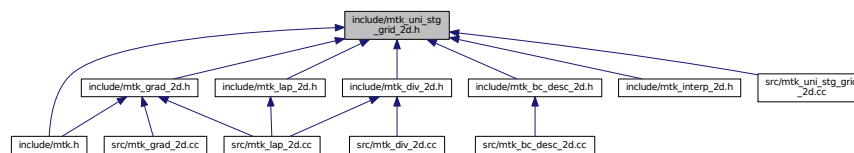
Definition of an 2D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk_uni_stg_grid_2d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::UniStgGrid2D`
Uniform 2D Staggered Grid.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

17.47.1 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_2d.h](#).

17.48 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
00079 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 =
mtk::SCALAR);
00114
00116     ~UniStgGrid2D();
00117
00123     Real *discrete_domain_x();
00124
00130     Real *discrete_domain_y();
00131
00137     Real *discrete_field();
00138
00146     FieldNature nature() const;
00147
00153     Real west_bndy() const;
00154
00160     Real east_bndy() const;
00161
00167     int num_cells_x() const;
00168
00174     Real delta_x() const;
00175
00181     Real south_bndy() const;
00182
00188     Real north_bndy() const;
00189
00195     int num_cells_y() const;
00196
00202     Real delta_y() const;
00203
00209     void BindScalarField(Real (*ScalarField)(Real xx, Real yy));
00210
00225     void BindVectorField(Real (*VectorFieldPComponent)(Real xx,
Real yy),
                                Real (*VectorFieldQComponent)(Real xx,Real yy));
00226
00227     bool WriteToFile(std::string filename,
00240                     std::string space_name_x,
00241                     std::string space_name_y,
00242                     std::string field_name);
00243
00244 private:
00245     void BindVectorFieldPComponent(
00258         Real (*VectorFieldPComponent)(Real xx, Real yy));
00259
00260     void BindVectorFieldQComponent(
00273         Real (*VectorFieldQComponent)(Real xx, Real yy));
00274
00275
00276
00277     std::vector<Real> discrete_domain_x_;
00278     std::vector<Real> discrete_domain_y_;
00279     std::vector<Real> discrete_field_;
00280
00281     FieldNature nature_;
00282
00283     Real west_bndy_;
00284     Real east_bndy_;
00285     int num_cells_x_;
00286     Real delta_x_;
00287
00288     Real south_bndy_;
00289     Real north_bndy_;
00290     int num_cells_y_;
00291     Real delta_y_;
00292 };
00293 }
00294 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_

```

17.49 Makefile.inc File Reference

17.50 Makefile.inc

```

00001 # Makefile setup file for MTK.
00002
00003 SHELL := /bin/bash
00004
00005 # Please set the following variables up:
00006
00007 #   1. Absolute path to base directory of the MTK.
00008 #   _____
00009
00010 BASE = /home/esanchez/Dropbox/MTK
00011
00012 #   2. The machine (platform) identifier and required machine precision.
00013 #   _____
00014
00015 # Options are:
00016 # - LINUX: A LINUX box installation.
00017 # - OSX: Uses OS X optimized solvers.
00018
00019 PLAT = LINUX
00020
00021 # Options are:
00022 # - SINGLE: Use 4 B floating point numbers.
00023 # - DOUBLE: Use 8 B floating point numbers.
00024
00025 PRECISION = DOUBLE
00026
00027 #   3. Optimized solvers and operations by means of ATLAS in Linux?
00028 #   _____
00029
00030 # If you have selected OSX in step 1, then you don't need to worry about this.
00031
00032 # Options are ON xor OFF:
00033
00034 ATL_OPT = OFF
00035
00036 #   4. Paths to dependencies (header files for compiling).
00037 #   _____
00038
00039 # GLPK include path (soon to go):
00040
00041 GLPK_INC = $(HOME)/Libraries/glpk-4.35/include
00042
00043 # Linux: If ATLAS optimization is ON, users should only provide the path to
00044 # ATLAS:
00045
00046 ATLAS_INC = $(HOME)/Libraries/ATLAS_3.8.4-CORE/include
00047
00048 # OS X: Do nothing.
00049
00050 #   5. Paths to dependencies (archive files for (static) linking).
00051 #   _____
00052
00053 # GLPK linking path (soon to go):
00054
00055 GLPK_LIB = $(HOME)/Libraries/glpk-4.35/lib/lib64/libglpk.a
00056
00057 # If optimization is OFF, then provide the paths for:
00058
00059 BLAS_LIB = $(HOME)/Libraries/BLAS-3.5.0/libblas.a
00060 LAPACK_LIB = $(HOME)/Libraries/lapack-3.5.0/liblapack.a
00061
00062 # WARNING: Vendor libraries should be used whenever they are available.
00063
00064 # However, if optimization is ON, please provide the path the ATLAS' archive:
00065
00066 ATLAS_LIB = $(HOME)/Libraries/ATLAS_3.8.4-CORE/ATLAS_3.8.4-BUILD-Citadel/lib
00067
00068 #   6. Compiler and its flags.
00069 #   _____
00070
00071 CC = g++
00072
00073 # Debug Level. Options are:

```

```

00074 # 0. NO debug at all NOR any run-time checks... be cautious!
00075 # 1. Verbose (execution messages) AND run-time checks.
00076 # 2. Level 1 plus intermediate scalar-valued results.
00077 # 3. Level 2 plus intermediate array-valued results.
00078
00079 DEBUG_LEVEL = 3
00080
00081 # Flags recommended for release code:
00082
00083 CCFLAGS = -Wall -O3
00084
00085 # Flags recommended for debugging code:
00086
00087 CCFLAGS = -Wall -g
00088
00089 # 7. Archiver, its flags, and ranlib:
00090 # _____
00091
00092 ARCH      = ar
00093 ARCHFLAGS = cr
00094
00095 # If your system does not have "ranlib" then set: "RANLIB = echo":
00096
00097 RANLIB = echo
00098
00099 # But, if possible:
00100
00101 RANLIB = ranlib
00102
00103 # 8. Valgrind's memcheck options (optional):
00104 # _____
00105
00106 MEMCHECK_OPTS = -v --tool=memcheck --leak-check=full --show-leak-kinds=all \
00107 --track-origins=yes --freelist-vol=20000000
00108
00109 # Done! User, please, do not mess with the definitions from this point on.
00110
00111 # _____
00112 # _____
00113 # _____
00114
00115 # MTK-related.
00116 # _____
00117
00118 SRC      = $(BASE)/src
00119 INCLUDE  = $(BASE)/include
00120 LIB      = $(BASE)/lib
00121 MTK_LIB  = $(LIB)/libmtk.a
00122 TESTS    = $(BASE)/tests
00123 EXAMPLES = $(BASE)/examples
00124
00125 # Compiling-related.
00126 # _____
00127
00128 CCFLAGS += -std=c++11 -fPIC -DMTK_DEBUG_LEVEL=$(DEBUG_LEVEL) -I$(INCLUDE) -c
00129
00130 ifeq ($(PRECISION),DOUBLE)
00131     CCFLAGS += -DMTK_PRECISION_DOUBLE
00132 else
00133     CCFLAGS += -DMTK_PRECISION_SINGLE
00134 endif
00135
00136 # Only the GLPK is included because the other dependencies are coded in Fortran.
00137
00138 ifeq ($(ATL_OPT),ON)
00139     CCFLAGS += -I$(GLPK_INC) $(ATLAS_INC)
00140 else
00141     CCFLAGS += -I$(GLPK_INC)
00142 endif
00143
00144 # Linking-related.
00145 # _____
00146
00147 NOOPT_LIBS = $(LAPACK_LIB) $(BLAS_LIB) -lm $(GLPK_LIB) -lstdc++
00148
00149 OPT_LIBS    = -L$(ATLAS_LIB) -latlas -llapack -lblas -lm -latlas -lstdc++
00150
00151 ifeq ($(PLAT),OSX)
00152     LINKER = g++
00153     LINKER += -framework Accelerate $(GLPK_LIB) $(MTK_LIB)
00154 else

```

```

00155 ifeq ($(ATL_OPT),ON)
00156     LINKER = g++
00157     LIBS = $(MTK_LIB)
00158     LIBS += $(OPT_LIBS)
00159 else
00160     LINKER = gfortran
00161     LIBS = $(MTK_LIB)
00162     LIBS += $(NOOPT_LIBS)
00163 endif
00164 endif
00165
00166 # Documentation-related.
00167 # -----
00168
00169 DOCGEN = doxygen
00170 DOCFilename = doc_config.dxcf
00171 DOC = $(BASE)/doc
00172 DOCFILE = $(BASE)/$(DOCFilename)

```

17.51 README.md File Reference

17.52 README.md

```

00001 # The Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu**
00004 -----
00005
00006 ## 1. Description
00007
00008 We define numerical methods that are based on discretizations preserving the
00009 properties of their continuum counterparts to be **mimetic**.
00010
00011 The **Mimetic Methods Toolkit (MTK)** is a C++ library for mimetic numerical
00012 methods. It is arranged as a set of classes for **mimetic quadratures**,
00013 **mimetic interpolation**, and **mimetic finite differences** methods for the
00014 numerical solution of ordinary and partial differential equations.
00015
00016 An older version of this library is available outside of GitHub... just email me
00017 about it, and you can have it... it is ugly, yet it is functional and more
00018 complete.
00019 -----
00020
00021 ## 2. Dependencies
00022
00023 This README assumes all of these dependencies are installed in the following
00024 folder:
00025 ```
00026 ```
00027 $(HOME)/Libraries/
00028 ```
00029
00030 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00031 routines for the internal computation on some of the layers. However, ATLAS
00032 requires both BLAS and LAPACK in order to create their optimized distributions.
00033 Therefore, the following dependencies tree arises:
00034
00035 ### For Linux:
00036
00037 1. LAPACK - Available from: http://www.netlib.org/lapack/
00038 1. BLAS - Available from: http://www.netlib.org/blas/
00039
00040 2. GLPK - Available from: https://www.gnu.org/software/glpk/
00041
00042 3. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00043 1. LAPACK - Available from: http://www.netlib.org/lapack/
00044 1. BLAS - Available from: http://www.netlib.org/blas
00045
00046 4. (Optional) Valgrind - Available from: http://valgrind.org/
00047
00048 5. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00049
00050 ### For OS X:
00051
00052 1. GLPK - Available from: https://www.gnu.org/software/glpk/

```



```
00053
00054
00055 ## 3. Installation
00056
00057 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00058
00059 The following steps are required to build and test the MTK. Please use the
00060 accompanying 'Makefile.inc' file, which should provide a solid template to
00061 start with. The following command provides help on the options for make:
00062
00063 ```
00064 $ make help
00065 -----
00066 Makefile for the MTK.
00067
00068 Options are:
00069 - all: builds the library, the tests, and examples.
00070 - mtklib: builds the library.
00071 - test: builds the test files.
00072 - example: builds the examples.
00073
00074 - testall: runs all the tests.
00075
00076 - gendoc: generates the documentation for the library.
00077
00078 - clean: cleans all the generated files.
00079 - cleanlib: cleans the generated archive and object files.
00080 - cleantest: cleans the generated tests executables.
00081 - cleanexample: cleans the generated examples executables.
00082 -----
00083 ```
00084
00085 ### PART 2. BUILD THE LIBRARY.
00086
00087 ```
00088 $ make
00089 ```
00090
00091 If successful you'll read (before building the tests and examples):
00092
00093 ```
00094 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00095 ```
00096
00097 Examples and tests will also be built.
00098
00099
00100 ## 4. Frequently Asked Questions
00101
00102 Q: Why haven't you guys implemented GBS to build the library?
00103 A: I'm on it as we speak! ;)
00104
00105 Q: Is there any main reference when it comes to the theory on Mimetic Methods?
00106 A: Yes! Check: http://www.csrc.sdsu.edu/mimetic-book
00107
00108 Q: Do I need to generate the documentation myself?
00109 A: You can if you want to... but if you DO NOT want to, just go to our website.
00110
00111
00112 ## 5. Contact, Support, and Credits
00113
00114 The MTK is developed by researchers and adjuncts to the
00115 [Computational Science Research Center (CSRC)](http://www.csrc.sdsu.edu/)
00116 at [San Diego State University (SDSU)](http://www.sdsu.edu/).
00117
00118 Developers are members of:
00119
00120 1. Mimetic Numerical Methods Research and Development Group.
00121 2. Computational Geoscience Research and Development Group.
00122 3. Ocean Modeling Research and Development Group.
00123
00124 Currently the developers are:
00125
00126 - *Eduardo J. Sanchez, Ph.D. - esanchez@mail.sdsu.edu* - @ejspeiro
00127 - Jose E. Castillo, Ph.D. - jcastillo@mail.sdsu.edu
00128 - Guillermo F. Miranda, Ph.D. - unigrav@hotmail.com
00129 - Christopher P. Paolini, Ph.D. - paolini@engineering.sdsu.edu
00130 - Angel Boada.
00131 - Johnny Corbino.
00132 - Raul Vargas-Navarro.
00133
```

```

00134 Finally, please feel free to contact me with suggestions or corrections:
00135
00136 **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu** - @ejspeiro
00137
00138 Thanks and happy coding!

```

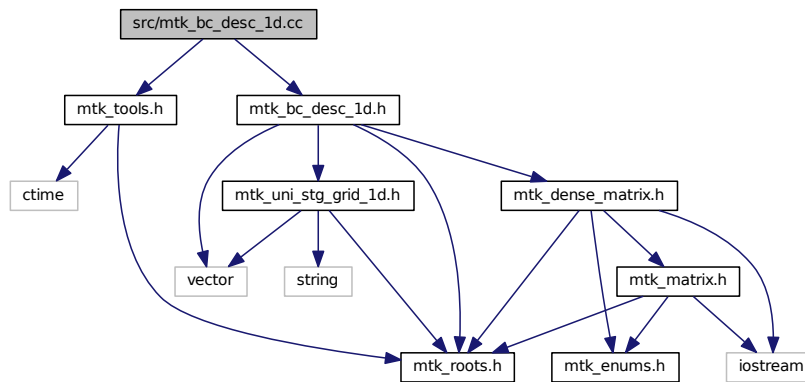
17.53 src/mtk_bc_desc_1d.cc File Reference

Enforces boundary conditions in either the operator or the grid.

```
#include "mtk_tools.h"
```

```
#include "mtk_bc_desc_1d.h"
```

Include dependency graph for mtk_bc_desc_1d.cc:



17.53.1 Detailed Description

This class presents an interface for the user to specify boundary conditions on 1D mimetic operators and the grids they are acting on.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_bc_desc_1d.cc](#).

17.54 mtk_bc_desc_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 "mtk_tools.h"
00058
00059 #include "mtk_bc_desc_1d.h"
00060
00061 void mtk::BCDesc1D::ImposeOnOperatorMatrix(
00062     mtk::DenseMatrix &matrix,
00063     const std::vector<mtk::Real> &west,
00064     const std::vector<mtk::Real> &east) {
00065     #if MTK_DEBUG_LEVEL > 0
00066         mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00067         mtk::Tools::Prevent(west.size() > (unsigned int) matrix.
00068             num_cols(),
00069             __FILE__, __LINE__, __func__);
00070         mtk::Tools::Prevent(east.size() > (unsigned int) matrix.
00071             num_cols(),
00072             __FILE__, __LINE__, __func__);
00073     #endif
00074     for (unsigned int ii = 0; ii < west.size(); ++ii) {
00075         matrix.SetValue(0, ii, west[ii]);
00076     }
00077     for (unsigned int ii = 0; ii < east.size(); ++ii) {
00078         matrix.SetValue(matrix.num_rows() - 1,
00079             matrix.num_cols() - 1 - ii,
00080             east[ii]);
00081     }
00082 }
00083
00084 void mtk::BCDesc1D::ImposeOnGrid(mtk::UniStgGrid1D &grid,
00085     const mtk::Real &omega,
00086     const mtk::Real &epsilon) {
00087     #if MTK_DEBUG_LEVEL > 0
00088         mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00089     #endif
00090     grid.discrete_field_u()[0] = omega;
00091     grid.discrete_field_u()[grid.num_cells_x() + 2 - 1] = epsilon;

```

```
00103 }
```

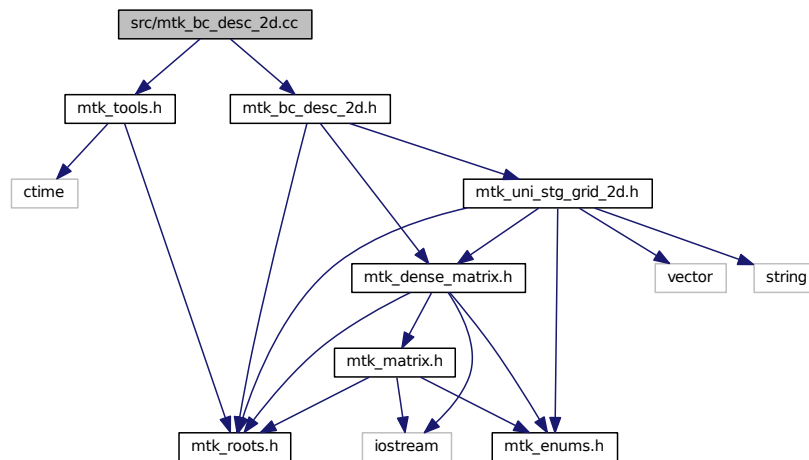
17.55 src/mtk_bc_desc_2d.cc File Reference

Enforces boundary conditions in either the operator or the grid.

```
#include "mtk_tools.h"
```

```
#include "mtk_bc_desc_2d.h"
```

Include dependency graph for mtk_bc_desc_2d.cc:



17.55.1 Detailed Description

This class implements an interface for the user to specify boundary conditions on 2D mimetic operators and the grids they are acting on.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_bc_desc_2d.cc](#).

17.56 mtk_bc_desc_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 "mtk_tools.h"
00058
00059 #include "mtk_bc_desc_2d.h"
00060
00061 void mtk::BCDesc2D::ImposeOnOperatorMatrix(const
    mtk::DenseMatrix &matrix,
00062                                     mtk::Real (*west)(int ii, int jj),
00063                                     mtk::Real (*east)(int ii, int jj),
00064                                     mtk::Real (*south)(int ii, int jj),
00065                                     mtk::Real (*north)(int ii, int jj)) {
00066
00067     #if MTK_DEBUG_LEVEL > 0
00068     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00069     mtk::Tools::Prevent(west == nullptr, __FILE__, __LINE__, __func__);
00070     mtk::Tools::Prevent(east == nullptr, __FILE__, __LINE__, __func__);
00071     mtk::Tools::Prevent(south == nullptr, __FILE__, __LINE__, __func__);
00072     mtk::Tools::Prevent(north == nullptr, __FILE__, __LINE__, __func__);
00073     #endif
00074
00075 }
00076
00077
00078 void mtk::BCDesc2D::ImposeOnGrid(const
    mtk::UniStgGrid2D &grid,
00079     mtk::Real (*west)(mtk::Real xx, mtk::Real yy),
00080     mtk::Real (*east)(mtk::Real xx, mtk::Real yy),
00081     mtk::Real (*south)(mtk::Real xx, mtk::Real yy),
00082     mtk::Real (*north)(mtk::Real xx, mtk::Real yy)) {
00083
00084     #if MTK_DEBUG_LEVEL > 0
00085     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00086     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00087     mtk::Tools::Prevent(west == nullptr, __FILE__, __LINE__, __func__);
00088     mtk::Tools::Prevent(east == nullptr, __FILE__, __LINE__, __func__);
00089     mtk::Tools::Prevent(south == nullptr, __FILE__, __LINE__, __func__);
00090     mtk::Tools::Prevent(north == nullptr, __FILE__, __LINE__, __func__);
00091     #endif
00092
00093 }

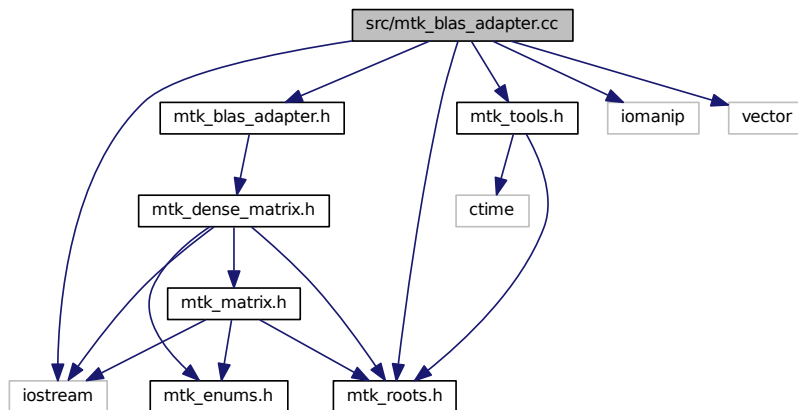
```

17.57 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)

17.57.1 Detailed Description

This class contains a collection of static classes, 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.cc](#).

17.58 mtk_blas_adapter.cc

```
00001
00024 /*
00025 Copyright (C) 2015, Computational Science Research Center, San Diego State
00026 University. All rights reserved.
00027
00028 Redistribution and use in source and binary forms, with or without modification,
00029 are permitted provided that the following conditions are met:
00030
00031 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00032 and a copy of the modified files should be reported once modifications are
00033 completed, unless these modifications are made through the project's GitHub
00034 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00035 should be developed and included in any deliverable.
00036
00037 2. Redistributions of source code must be done through direct
00038 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00039
00040 3. Redistributions in binary form must reproduce the above copyright notice,
00041 this list of conditions and the following disclaimer in the documentation and/or
00042 other materials provided with the distribution.
00043
00044 4. Usage of the binary form on proprietary applications shall require explicit
00045 prior written permission from the the copyright holders, and due credit should
00046 be given to the copyright holders.
00047
00048 5. Neither the name of the copyright holder nor the names of its contributors
00049 may be used to endorse or promote products derived from this software without
00050 specific prior written permission.
00051
00052 The copyright holders provide no reassurances that the source code provided does
00053 not infringe any patent, copyright, or any other intellectual property rights of
00054 third parties. The copyright holders disclaim any liability to any recipient for
00055 claims brought against recipient by any third party for infringement of that
00056 parties intellectual property rights.
00057
00058 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00059 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00060 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00061 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00062 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00063 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00064 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00065 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00066 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <vector>
00074
00075 #include "mtk_roots.h"
00076 #include "mtk_tools.h"
00077 #include "mtk_blas_adapter.h"
00078
```

```

00079 namespace mtk {
00080
00081 extern "C" {
00082
00083 #ifdef MTK_PRECISION_DOUBLE
00084
00097 double dnrms2__(int *n, double *x, int *incx);
00098 #else
00099
00112 float snrms2__(int *n, float *x, int *incx);
00113 #endif
00114
00115 #ifdef MTK_PRECISION_DOUBLE
00116
00135 void daxpy__(int *n, double *da, double *dx, int *incx, double *dy, int *incy);
00136 #else
00137
00156 void saxpy__(int *n, float *sa, float *sx, int *incx, float *sy, int *incy);
00157 #endif
00158
00159 #ifdef MTK_PRECISION_DOUBLE
00160
00188 void dgemv__(char *trans,
00189              int *m,
00190              int *n,
00191              double *alpha,
00192              double *a,
00193              int *lda,
00194              double *x,
00195              int *incx,
00196              double *beta,
00197              double *y,
00198              int *incy);
00199 #else
00200
00228 void sgemv__(char *trans,
00229              int *m,
00230              int *n,
00231              float *alpha,
00232              float *a,
00233              int *lda,
00234              float *x,
00235              int *incx,
00236              float *beta,
00237              float *y,
00238              int *incy);
00239 #endif
00240
00241 #ifdef MTK_PRECISION_DOUBLE
00242
00267 void dgemm__(char *transa,
00268              char* transb,
00269              int *m,
00270              int *n,
00271              int *k,
00272              double *alpha,
00273              double *a,
00274              int *lda,
00275              double *b,
00276              int *ldb,
00277              double *beta,
00278              double *c,
00279              int *ldc);
00280 }
00281 #else
00282
00307 void sgemm__(char *transa,
00308              char* transb,
00309              int *m,
00310              int *n,
00311              int *k,
00312              double *alpha,
00313              double *a,
00314              int *lda,
00315              double *b, aamm
00316              int *ldb,
00317              double *beta,
00318              double *c,
00319              int *ldc);
00320 }
00321 #endif

```



```

00322 }
00323
00324 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00325
00326     #if MTK_DEBUG_LEVEL > 0
00327     mtk::Tools::Prevent(in_length <= 0, __FILE__, __LINE__, __func__);
00328     #endif
00329
00330     int incx{1}; // Increment for the elements of xx. ix >= 0.
00331
00332     #ifdef MTK_PRECISION_DOUBLE
00333     return dnorm2(&in_length, in, &incx);
00334     #else
00335     return snrm2(&in_length, in, &incx);
00336     #endif
00337 }
00338
00339 void mtk::BLASAdapter::RealAXPY(mtk::Real alpha,
00340                                 mtk::Real *xx,
00341                                 mtk::Real *yy,
00342                                 int &in_length) {
00343
00344     #if MTK_DEBUG_LEVEL > 0
00345     mtk::Tools::Prevent(xx == nullptr, __FILE__, __LINE__, __func__);
00346     mtk::Tools::Prevent(yy == nullptr, __FILE__, __LINE__, __func__);
00347     #endif
00348
00349     int incx{1}; // Increment for the elements of xx. ix >= 0.
00350
00351     #ifdef MTK_PRECISION_DOUBLE
00352     daxpy(&in_length, &alpha, xx, &incx, yy, &incx);
00353     #else
00354     saxpy(&in_length, &alpha, xx, &incx, yy, &incx);
00355     #endif
00356 }
00357
00358 mtk::Real mtk::BLASAdapter::RelNorm2Error(
00359     mtk::Real *computed,
00360     mtk::Real *known,
00361     int length) {
00362
00363     #if MTK_DEBUG_LEVEL > 0
00364     mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00365     mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00366     #endif
00367
00368     mtk::Real norm_2_computed(mtk::BLASAdapter::RealNRM2(known, length));
00369
00370     mtk::Real alpha{-mtk::kOne};
00371
00372     mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00373
00374     mtk::Real norm_2_difference(mtk::BLASAdapter::RealNRM2(computed,
00375     length));
00376
00377     return norm_2_difference/norm_2_computed;
00378 }
00379
00380 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00381                                     mtk::DenseMatrix &aa,
00382                                     mtk::Real *xx,
00383                                     mtk::Real &beta,
00384                                     mtk::Real *yy) {
00385
00386     // Make sure input matrices are row-major ordered.
00387
00388     if (aa.matrix_properties().ordering() ==
00389         mtk::COL_MAJOR) {
00390         aa.OrderRowMajor();
00391     }
00392
00393     char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00394
00395     int mm{aa.num_rows()}; // Rows of aa.
00396     int nn{aa.num_cols()}; // Columns of aa.
00397     int lda{(aa.matrix_properties()).ld()}; // Leading dimension.
00398     int incx{1}; // Increment of values in x.
00399     int incy{1}; // Increment of values in y.
00400
00401     std::swap(mm, nn);
00402     #ifdef MTK_PRECISION_DOUBLE

```

```

00400   dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00401         xx, &incx, &beta, yy, &incy);
00402   #else
00403   sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00404         xx, &incx, &beta, yy, &incy);
00405   #endif
00406   std::swap(mm,nn);
00407 }
00408
00409 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
    mtk::DenseMatrix &aa,
                                mtk::DenseMatrix &bb) {
00410
00411   #if MTK_DEBUG_LEVEL > 0
00412   mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00413     __FILE__, __LINE__, __func__);
00414   #endif
00415
00416   // Make sure input matrices are row-major ordered.
00417   if (aa.matrix_properties().ordering() ==
    mtk::COL_MAJOR) {
00420     aa.OrderRowMajor();
00421   }
00422   if (bb.matrix_properties().ordering() ==
    mtk::COL_MAJOR) {
00423     bb.OrderRowMajor();
00424   }
00425
00426   char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00427   char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00428
00429   int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00430   int nn{bb.num_cols()}; // Cols of bb and cols of cc.
00431   int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00432
00433   int cc_num_rows{mm}; // Rows of cc.
00434   int cc_num_cols{nn}; // Columns of cc.
00435
00436   int lda{std::max(1, kk)}; // Leading dimension of the aa matrix.
00437   int ldb{std::max(1, nn)}; // Leading dimension of the bb matrix.
00438   int ldc{std::max(1, mm)}; // Leading dimension of the cc matrix.
00439
00440   mtk::Real alpha{1.0}; // First scalar coefficient.
00441   mtk::Real beta{0.0}; // Second scalar coefficient.
00442
00443   mtk::DenseMatrix cc_col_maj_ord(cc_num_rows, cc_num_cols); // Output matrix.
00444
00445   cc_col_maj_ord.SetOrdering(mtk::COL_MAJOR);
00446
00447   #ifdef MTK_PRECISION_DOUBLE
00448   dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00449         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00450   #else
00451   sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00452         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00453   #endif
00454
00455   #if MTK_DEBUG_LEVEL > 0
00456   std::cout << "cc_col_maj_ord =" << std::endl;
00457   std::cout << cc_col_maj_ord << std::endl;
00458   #endif
00459
00460   cc_col_maj_ord.OrderRowMajor();
00461
00462   return cc_col_maj_ord;
00463 }

```

17.59 src/mtk_dense_matrix.cc File Reference

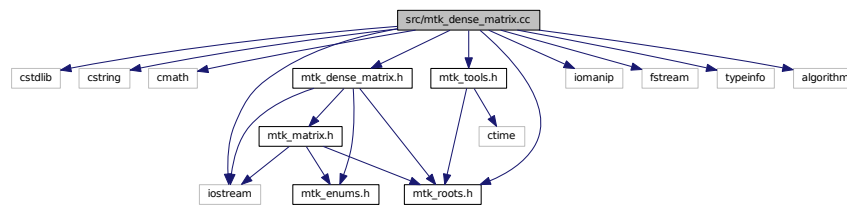
```
#include <cstdlib>
```

```

#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <typeinfo>
#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)`

17.60 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 <algorithm>
00070
00071 #include "mtk_roots.h"
00072 #include "mtk_dense_matrix.h"
00073 #include "mtk_tools.h"
00074
00075 namespace mtk {
00076
00077 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00078
00079     int mm{in.matrix_properties_.num_rows()}; // Auxiliary.
00080     int nn{in.matrix_properties_.num_cols()}; // Auxiliary.
00081
00082     if (in.matrix_properties_.ordering() ==
00083         mtk::COL_MAJOR) {
00084         std::swap(mm, nn);
00085     }
00086     for (int ii = 0; ii < mm; ii++) {
00087         int offset(ii*nn);
00088         for (int jj = 0; jj < nn; jj++) {
00089             mtk::Real value = in.data_[offset + jj];
00090             stream << std::setw(9) << value;
00091         }
00092         stream << std::endl;
00093     }
00094     if (in.matrix_properties_.ordering() ==
00095         mtk::COL_MAJOR) {
00096         std::swap(mm, nn);
00097     }
00098     return stream;
00099 }
00100 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
00101     mtk::DenseMatrix &in) {
00102
00103     if(this == &in) {
00104         return *this;
00105     }
00106     matrix_properties_.set_storage(in.
00107         matrix_properties_.storage());
00108     matrix_properties_.set_ordering(in.
00109         matrix_properties_.ordering());
00110
00111     auto aux = in.matrix_properties_.num_rows();
00112     matrix_properties_.set_num_rows(aux);
00113

```

```

00113     aux = in.matrix_properties().num_cols();
00114     matrix_properties_.set_num_cols(aux);
00115
00116     aux = in.matrix_properties().num_zero();
00117     matrix_properties_.set_num_zero(aux);
00118
00119     aux = in.matrix_properties().num_null();
00120     matrix_properties_.set_num_null(aux);
00121
00122     auto num_rows = matrix_properties_.num_rows();
00123     auto num_cols = matrix_properties_.num_cols();
00124
00125     delete [] data_;
00126
00127     try {
00128         data_ = new mtk::Real[num_rows*num_cols];
00129     } catch (std::bad_alloc &memory_allocation_exception) {
00130         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00131             std::endl;
00132         std::cerr << memory_allocation_exception.what() << std::endl;
00133     }
00134     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
num_cols);
00135
00136     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00137
00138     return *this;
00139 }
00140
00141 bool mtk::DenseMatrix::operator ==(const
DenseMatrix &in) {
00142
00143     bool ans{true};
00144
00145     auto mm = in.num_rows();
00146     auto nn = in.num_cols();
00147
00148     if (mm != matrix_properties_.num_rows() ||
00149         nn != matrix_properties_.num_cols()) {
00150         return false;
00151     }
00152
00153     for (int ii = 0; ii < mm && ans; ++ii) {
00154         for (int jj = 0; jj < nn && ans; ++jj) {
00155             ans = ans &&
00156                 abs(data_[ii*nn + jj] - in.data()[ii*nn + jj]) <
mtk::kDefaultTolerance;
00157         }
00158     }
00159     return ans;
00160 }
00161
00162 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00163
00164     matrix_properties_.set_storage(mtk::DENSE);
00165     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00166 }
00167
00168 mtk::DenseMatrix::DenseMatrix(const
mtk::DenseMatrix &in) {
00169
00170     matrix_properties_.set_storage(in.matrix_properties_.storage());
00171
00172     matrix_properties_.set_ordering(in.matrix_properties_.
ordering());
00173
00174     auto aux = in.matrix_properties_.num_rows();
00175     matrix_properties_.set_num_rows(aux);
00176
00177     aux = in.matrix_properties().num_cols();
00178     matrix_properties_.set_num_cols(aux);
00179
00180     aux = in.matrix_properties().num_zero();
00181     matrix_properties_.set_num_zero(aux);
00182
00183     aux = in.matrix_properties().num_null();
00184     matrix_properties_.set_num_null(aux);
00185
00186     auto num_rows = in.matrix_properties_.num_rows();
00187     auto num_cols = in.matrix_properties_.num_cols();
00188

```

```

00189     try {
00190         data_ = new mtk::Real[num_rows*num_cols];
00191     } catch (std::bad_alloc &memory_allocation_exception) {
00192         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00193             std::endl;
00194         std::cerr << memory_allocation_exception.what() << std::endl;
00195     }
00196     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00197
00198     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00199 }
00200
00201 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00202
00203     #if MTK_DEBUG_LEVEL > 0
00204     mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00205     mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00206     #endif
00207
00208     matrix_properties_.set_storage(mtk::DENSE);
00209     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00210     matrix_properties_.set_num_rows(num_rows);
00211     matrix_properties_.set_num_cols(num_cols);
00212
00213     try {
00214         data_ = new mtk::Real[num_rows*num_cols];
00215     } catch (std::bad_alloc &memory_allocation_exception) {
00216         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00217             std::endl;
00218         std::cerr << memory_allocation_exception.what() << std::endl;
00219     }
00220     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00221 }
00222
00223 mtk::DenseMatrix::DenseMatrix(const int &rank,
00224                               const bool &padded,
00225                               const bool &transpose) {
00226
00227     #if MTK_DEBUG_LEVEL > 0
00228     mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00229     #endif
00230
00231     int aux{}; // Used to control the padding.
00232
00233     if (padded) {
00234         aux = 1;
00235     }
00236
00237     matrix_properties_.set_storage(mtk::DENSE);
00238     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00239     matrix_properties_.set_num_rows(aux + rank + aux);
00240     matrix_properties_.set_num_cols(rank);
00241
00242     try {
00243         data_ = new mtk::Real[matrix_properties_.num_values()];
00244     } catch (std::bad_alloc &memory_allocation_exception) {
00245         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00246             std::endl;
00247         std::cerr << memory_allocation_exception.what() << std::endl;
00248     }
00249     memset(data_,
00250            mtk::kZero,
00251            sizeof(data_[0])*(matrix_properties_.num_values()));
00252
00253     for (auto ii = 0; ii < matrix_properties_.num_rows(); ++ii) {
00254         for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00255             data_[ii*matrix_properties_.num_cols() + jj] =
00256                 (ii == jj + aux)? mtk::kOne: mtk::kZero;
00257         }
00258     }
00259     if (transpose) {
00260         Transpose();
00261     }
00262 }
00263
00264 mtk::DenseMatrix::DenseMatrix(const mtk::Real *gen,
00265                               const int &gen_length,
00266                               const int &pro_length,
00267                               const bool &transpose) {
00268
00269     #if MTK_DEBUG_LEVEL > 0

```

```

00270 mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00271 mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00272 mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00273 #endif
00274
00275 matrix_properties_.set_storage(mtk::DENSE);
00276 matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00277 if (!transpose) {
00278     matrix_properties_.set_num_rows(gen_length);
00279     matrix_properties_.set_num_cols(pro_length);
00280 } else {
00281     matrix_properties_.set_num_rows(pro_length);
00282     matrix_properties_.set_num_cols(gen_length);
00283 }
00284
00285 int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00286 int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00287
00288 try {
00289     data_ = new mtk::Real[mm*nn];
00290 } catch (std::bad_alloc &memory_allocation_exception) {
00291     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00292         std::endl;
00293     std::cerr << memory_allocation_exception.what() << std::endl;
00294 }
00295 memset(data_, mtk::kZero, sizeof(data_[0])*mm*nn);
00296
00297 if (!transpose) {
00298     for (auto ii = 0; ii < mm; ii++) {
00299         for (auto jj = 0; jj < nn; jj++) {
00300             data_[ii*nn + jj] = pow(gen[ii], (double) jj);
00301         }
00302     }
00303 } else {
00304     for (auto ii = 0; ii < mm; ii++) {
00305         for (auto jj = 0; jj < nn; jj++) {
00306             data_[ii*nn + jj] = pow(gen[jj], (double) ii);
00307         }
00308     }
00309 }
00310 }
00311
00312 mtk::DenseMatrix::~DenseMatrix() {
00313     delete[] data_;
00314     data_ = nullptr;
00315 }
00316
00317 mtk::Matrix mtk::DenseMatrix::matrix_properties() const {
00318     return matrix_properties_;
00319 }
00320
00321 void mtk::DenseMatrix::SetOrdering(
    mtk::MatrixOrdering oo) {
00322
00323     #if MTK_DEBUG_LEVEL > 0
00324     mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
00325         mtk::COL_MAJOR),
00326         __FILE__, __LINE__, __func__);
00327     #endif
00328
00329     matrix_properties_.set_ordering(oo);
00330 }
00331
00332 int mtk::DenseMatrix::num_rows() const {
00333     return matrix_properties_.num_rows();
00334 }
00335
00336 int mtk::DenseMatrix::num_cols() const {
00337     return matrix_properties_.num_cols();
00338 }
00339
00340 mtk::Real* mtk::DenseMatrix::data() const {
00341     return data_;
00342 }
00343
00344 mtk::Real mtk::DenseMatrix::GetValue(

```

```

00349     const int &mm,
00350     const int &nn) const {
00351
00352     #if MTK_DEBUG_LEVEL > 0
00353     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00354     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00355     #endif
00356
00357     return data_[mm*matrix_properties_.num_cols() + nn];
00358 }
00359
00360 void mtk::DenseMatrix::SetValue(
00361     const int &mm,
00362     const int &nn,
00363     const mtk::Real &val) {
00364
00365     #if MTK_DEBUG_LEVEL > 0
00366     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00367     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00368     #endif
00369
00370     data_[mm*matrix_properties_.num_cols() + nn] = val;
00371 }
00372
00373 void mtk::DenseMatrix::Transpose() {
00374
00375     mtk::Real *data_transposed{}; // Buffer.
00376
00377     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00378     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00379
00380     try {
00381         data_transposed = new mtk::Real[mm*nn];
00382     } catch (std::bad_alloc &memory_allocation_exception) {
00383         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00384             std::endl;
00385         std::cerr << memory_allocation_exception.what() << std::endl;
00386     }
00387
00388     memset(data_transposed,
00389         mtk::kZero,
00390         sizeof(data_transposed[0])*mm*nn);
00391
00392     // Assign the values to their transposed position.
00393     for (auto ii = 0; ii < mm; ++ii) {
00394         for (auto jj = 0; jj < nn; ++jj) {
00395             data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00396         }
00397     }
00398
00399     // Swap pointers.
00400     auto tmp = data_; // Temporal holder.
00401     data_ = data_transposed;
00402     delete [] tmp;
00403     tmp = nullptr;
00404
00405     matrix_properties_.set_num_rows(nn);
00406     matrix_properties_.set_num_cols(mm);
00407 }
00408
00409 void mtk::DenseMatrix::OrderRowMajor() {
00410
00411     if (matrix_properties_.ordering() == mtk::COL_MAJOR) {
00412
00413         mtk::Real *data_transposed{}; // Buffer.
00414
00415         int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00416         int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00417
00418         try {
00419             data_transposed = new mtk::Real[mm*nn];
00420         } catch (std::bad_alloc &memory_allocation_exception) {
00421             std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00422                 std::endl;
00423             std::cerr << memory_allocation_exception.what() << std::endl;
00424         }
00425
00426         memset(data_transposed,
00427             mtk::kZero,
00428             sizeof(data_transposed[0])*mm*nn);
00429
00430     }
00431 }

```



```

00432     // Assign the values to their transposed position.
00433     std::swap(mm, nn);
00434     for (auto ii = 0; ii < mm; ++ii) {
00435         for (auto jj = 0; jj < nn; ++jj) {
00436             data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00437         }
00438     }
00439     std::swap(mm, nn);
00440
00441     // Swap pointers.
00442     auto tmp = data_; // Temporal holder.
00443     data_ = data_transposed;
00444     delete [] tmp;
00445     tmp = nullptr;
00446
00447     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00448 }
00449 }
00450
00451 void mtk::DenseMatrix::OrderColMajor() {
00452
00453     if (matrix_properties_.ordering() == ROW_MAJOR) {
00454
00455         mtk::Real *data_transposed{}; // Buffer.
00456
00457         int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00458         int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00459
00460         try {
00461             data_transposed = new mtk::Real[mm*nn];
00462         } catch (std::bad_alloc &memory_allocation_exception) {
00463             std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00464                 std::endl;
00465             std::cerr << memory_allocation_exception.what() << std::endl;
00466         }
00467         memset(data_transposed,
00468             mtk::kZero,
00469             sizeof(data_transposed[0])*mm*nn);
00470
00471         // Assign the values to their transposed position.
00472         for (auto ii = 0; ii < mm; ++ii) {
00473             for (auto jj = 0; jj < nn; ++jj) {
00474                 data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00475             }
00476         }
00477
00478         // Swap pointers.
00479         auto tmp = data_; // Temporal holder.
00480         data_ = data_transposed;
00481         delete [] tmp;
00482         tmp = nullptr;
00483
00484         matrix_properties_.set_ordering(mtk::COL_MAJOR);
00485     }
00486 }
00487 }
00488 }
00489
00490 mtk::DenseMatrix mtk::DenseMatrix::Kron(const
    mtk::DenseMatrix &aa,
                                const mtk::DenseMatrix &bb) {
00491
00492     int row_offset{}; // Offset for rows.
00493     int col_offset{}; // Offset for rows.
00494
00495     mtk::Real aa_factor{}; // Used in computation.
00496
00497     // Auxiliary variables:
00498     auto aux1 = aa.matrix_properties_.num_rows()*bb.
matrix_properties_.num_rows();
00499     auto aux2 = aa.matrix_properties_.num_cols()*bb.
matrix_properties_.num_cols();
00500
00501     mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00502
00503     int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00504
00505     auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00506     auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00507     auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00508     auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00509
00510

```

```

00511     for (auto ii = 0; ii < mm; ++ii) {
00512         row_offset = ii*pp;
00513         for (auto jj = 0; jj < nn; ++jj) {
00514             col_offset = jj*qq;
00515             aa_factor = aa.data_[ii*nn + jj];
00516             for (auto ll = 0; ll < pp; ++ll) {
00517                 for (auto oo = 0; oo < qq; ++oo) {
00518                     auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00519                     output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00520                 }
00521             }
00522         }
00523     }
00524
00525     output.matrix_properties_.set_storage(mtk::DENSE);
00526     output.matrix_properties_.set_ordering(
mtk::ROW_MAJOR);
00527
00528     return output;
00529 }
00530
00531 bool mtk::DenseMatrix::WriteToFile(std::string filename) {
00532
00533     std::ofstream output_dat_file; // Output file.
00534
00535     output_dat_file.open(filename);
00536
00537     if (!output_dat_file.is_open()) {
00538         return false;
00539     }
00540
00541     int mm{matrix_properties_.num_rows()};
00542     int nn{matrix_properties_.num_cols()};
00543
00544     for (int ii = 0; ii < mm; ++ii) {
00545         int offset{ii*nn};
00546         for (int jj = 0; jj < nn; ++jj) {
00547             output_dat_file << ii << ' ' << jj << ' ' << data_[offset + jj] <<
00548                 std::endl;
00549         }
00550     }
00551
00552     output_dat_file.close();
00553
00554     return true;
00555 }

```

17.61 src/mtk_div_1d.cc File Reference

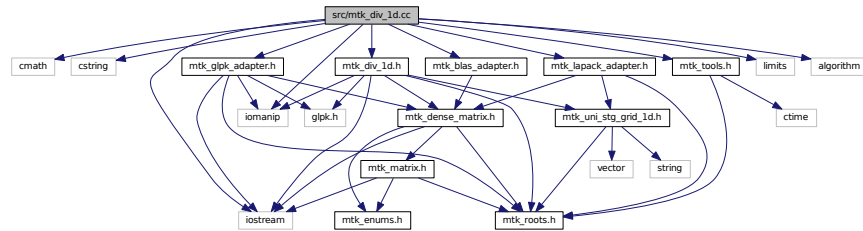
Implements the class Div1D.

```

#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_div_1d.h"

```

Include dependency graph for mtk_div_1d.cc:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

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

17.61.1 Detailed Description

This class implements a 1D divergence 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_div_1d.cc](#).

17.62 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 #include <limits>
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074
00075 #include "mtk_div_ld.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::DivLD &in) {
00080
00081     stream << "divergence_[0] = " << std::setw(9) << in.divergence_[0] <<
00082         std::endl;
00083
00084     stream << "divergence_[1:" << in.order_accuracy_ << "] = ";
00085     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00086         stream << std::setw(9) << in.divergence_[ii] << " ";
00087     }
00088     stream << std::endl;
00089
00090     if (in.order_accuracy_ > 2) {
00091
00092         stream << "divergence_[" << in.order_accuracy_ + 1 << ":" <<
00093             2*in.order_accuracy_ << "] = ";
00094         for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00095             order_accuracy_; ++ii) {
00096             stream << std::setw(9) << in.divergence_[ii] << " ";
00097         }
00098         stream << std::endl;
00099
00100         auto offset = (2*in.order_accuracy_ + 1);
00101         int mm{};
00102         for (auto ii = 0; ii < in.dim_null_; ++ii) {
00103             stream << "divergence_[" << offset + mm << ":" <<
00104                 offset + mm + in.num_bndy_coeffs_ - 1 << "] = ";
00105             for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {

```

```

00113         auto value = in.divergence_[offset + mm];
00114         stream << std::setw(9) << value << " ";
00115         ++mm;
00116     }
00117     stream << std::endl;
00118 }
00119 }
00120
00121 return stream;
00122 }
00123 }
00124
00125 mtk::Div1D::Div1D():
00126     order_accuracy_(mtk::kDefaultOrderAccuracy),
00127     dim_null_(),
00128     num_bndy_coeffs_(),
00129     divergence_length_(),
00130     minrow_(),
00131     row_(),
00132     coeffs_interior_(),
00133     prem_apps_(),
00134     weights_crs_(),
00135     weights_cbs_(),
00136     mim_bndy_(),
00137     divergence_(),
00138     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00139
00140 mtk::Div1D::Div1D(const Div1D &div):
00141     order_accuracy_(div.order_accuracy_),
00142     dim_null_(div.dim_null_),
00143     num_bndy_coeffs_(div.num_bndy_coeffs_),
00144     divergence_length_(div.divergence_length_),
00145     minrow_(div.minrow_),
00146     row_(div.row_),
00147     coeffs_interior_(div.coeffs_interior_),
00148     prem_apps_(div.prem_apps_),
00149     weights_crs_(div.weights_crs_),
00150     weights_cbs_(div.weights_cbs_),
00151     mim_bndy_(div.mim_bndy_),
00152     divergence_(div.divergence_),
00153     mimetic_threshold_(div.mimetic_threshold_) {}
00154
00155 mtk::Div1D::~Div1D() {
00156
00157     delete[] coeffs_interior_;
00158     coeffs_interior_ = nullptr;
00159
00160     delete[] prem_apps_;
00161     prem_apps_ = nullptr;
00162
00163     delete[] weights_crs_;
00164     weights_crs_ = nullptr;
00165
00166     delete[] weights_cbs_;
00167     weights_cbs_ = nullptr;
00168
00169     delete[] mim_bndy_;
00170     mim_bndy_ = nullptr;
00171
00172     delete[] divergence_;
00173     divergence_ = nullptr;
00174 }
00175
00176 bool mtk::Div1D::ConstructDiv1D(int order_accuracy,
00177                                 mtk::Real mimetic_threshold) {
00178
00179     #if MTK_DEBUG_LEVEL > 0
00180     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00181     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00182     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00183                         __FILE__, __LINE__, __func__);
00184
00185     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00186         std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00187     }
00188
00189     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00190     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00191     #endif
00192
00193     order_accuracy_ = order_accuracy;

```

```

00194     mimetic_threshold_ = mimetic_threshold;
00195
00197
00198     bool abort_construction = ComputeStencilInteriorGrid();
00199
00200     #if MTK_DEBUG_LEVEL > 0
00201     if (!abort_construction) {
00202         std::cerr << "Could NOT complete stage 1." << std::endl;
00203         std::cerr << "Exiting..." << std::endl;
00204         return false;
00205     }
00206     #endif
00207
00208     // At this point, we already have the values for the interior stencil stored
00209     // in the coeffs_interior_ array.
00210
00211     // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00212     // approximation at the boundary, thus it has no weights. For this case, the
00213     // dimension of the null-space of the Vandermonde matrices used to compute the
00214     // approximating coefficients at the boundary is 0. Ergo, we compute this
00215     // number first and then decide if we must compute anything at the boundary.
00216
00217     dim_null_ = order_accuracy_/2 - 1;
00218
00219     if (dim_null_ > 0) {
00220
00221         #ifdef MTK_PRECISION_DOUBLE
00222         num_bndy_coeffs_ = (int) (3.0*(mtk::Real) order_accuracy_)/2.0);
00223         #else
00224         num_bndy_coeffs_ = (int) (3.0f*(mtk::Real) order_accuracy_)/2.0f);
00225         #endif
00226
00227
00228         // For this we will follow recommendations given in:
00229         //
00230         // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00231         //
00232         // We will compute the QR Factorization of the transpose, as in the
00233         // following (MATLAB) pseudo-code:
00234         //
00235         // [Q,R] = qr(V'); % Full QR as defined in
00236         // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00237         //
00238         // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00239         //
00240         // However, given the nature of the Vandermonde matrices we've just
00241         // computed, they all posses the same null-space. Therefore, we impose the
00242         // convention of computing the null-space of the first Vandermonde matrix
00243         // (west boundary).
00244
00245         abort_construction = ComputeRationalBasisNullSpace();
00246
00247
00248         #if MTK_DEBUG_LEVEL > 0
00249         if (!abort_construction) {
00250             std::cerr << "Could NOT complete stage 2.1." << std::endl;
00251             std::cerr << "Exiting..." << std::endl;
00252             return false;
00253         }
00254         #endif
00255
00256
00257         abort_construction = ComputePreliminaryApproximations();
00258
00259
00260         #if MTK_DEBUG_LEVEL > 0
00261         if (!abort_construction) {
00262             std::cerr << "Could NOT complete stage 2.2." << std::endl;
00263             std::cerr << "Exiting..." << std::endl;
00264             return false;
00265         }
00266         #endif
00267
00268
00269         abort_construction = ComputeWeights();
00270
00271
00272         #if MTK_DEBUG_LEVEL > 0
00273         if (!abort_construction) {
00274             std::cerr << "Could NOT complete stage 2.3." << std::endl;
00275             std::cerr << "Exiting..." << std::endl;
00276             return false;
00277         }
00278         #endif

```

```

00279
00281
00282     abort_construction = ComputeStencilBoundaryGrid();
00283
00284     #if MTK_DEBUG_LEVEL > 0
00285     if (!abort_construction) {
00286         std::cerr << "Could NOT complete stage 2.4." << std::endl;
00287         std::cerr << "Exiting..." << std::endl;
00288         return false;
00289     }
00290     #endif
00291
00292 } // End of: if (dim_null_ > 0);
00293
00295
00296 // Once we have the following three collections of data:
00297 //   (a) the coefficients for the interior,
00298 //   (b) the coefficients for the boundary (if it applies),
00299 //   (c) and the weights (if it applies),
00300 // we will store everything in the output array:
00301
00302 abort_construction = AssembleOperator();
00303
00304 #if MTK_DEBUG_LEVEL > 0
00305 if (!abort_construction) {
00306     std::cerr << "Could NOT complete stage 3." << std::endl;
00307     std::cerr << "Exiting..." << std::endl;
00308     return false;
00309 }
00310 #endif
00311
00312 return true;
00313 }
00314
00315 int mtk::Div1D::num_bndy_coeffs() const {
00316
00317     return num_bndy_coeffs_;
00318 }
00319
00320 mtk::Real *mtk::Div1D::coeffs_interior() const {
00321
00322     return coeffs_interior_;
00323 }
00324
00325 mtk::Real *mtk::Div1D::weights_crs() const {
00326
00327     return weights_crs_;
00328 }
00329
00330 mtk::Real *mtk::Div1D::weights_cbs() const {
00331
00332     return weights_cbs_;
00333 }
00334
00335
00336 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00337
00338     mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00339
00340     auto counter = 0;
00341     for (auto ii = 0; ii < dim_null_; ++ii) {
00342         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00343             xx.SetValue(ii,jj, divergence_[2*order_accuracy_ + 1 + counter]);
00344             counter++;
00345         }
00346     }
00347
00348     return xx;
00349 }
00350
00351 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(const
UniStgGrid1D &grid) {
00352
00353     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00354
00355     #if MTK_DEBUG_LEVEL > 0
00356     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00357     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00358     #endif
00359
00360     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};

```

```

00361
00362 int dd_num_rows = nn + 2;
00363 int dd_num_cols = nn + 1;
00364 int elements_per_row = num_bndy_coeffs_;
00365 int num_extra_rows = dim_null_;
00366
00367 // Output matrix featuring sizes for divergence operators.
00368 mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00369
00371
00372 auto ee_index = 0;
00373 for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00374     auto cc = 0;
00375     for(auto jj = 0 ; jj < dd_num_rows; jj++) {
00376         if( cc >= elements_per_row) {
00377             out.SetValue(ii, jj, mtk::kZero);
00378         } else {
00379             out.SetValue(ii, jj, mim_bndy_[ee_index++]*inv_delta_x);
00380             cc++;
00381         }
00382     }
00383 }
00384
00386
00387 for (auto ii = num_extra_rows + 1;
00388     ii < dd_num_rows - num_extra_rows - 1; ii++) {
00389     auto jj = ii - num_extra_rows - 1;
00390     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00391         out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00392     }
00393 }
00394
00396
00397 ee_index = 0;
00398 for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--) {
00399     auto cc = 0;
00400     for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00401         if( cc >= elements_per_row) {
00402             out.SetValue(ii, jj, 0.0);
00403         } else {
00404             out.SetValue(ii, jj, -mim_bndy_[ee_index++]*inv_delta_x);
00405             cc++;
00406         }
00407     }
00408 }
00409
00410 return out;
00411 }
00412
00413 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00414
00416
00417     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00418
00419     try {
00420         pp = new mtk::Real[order_accuracy_];
00421     } catch (std::bad_alloc &memory_allocation_exception) {
00422         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00423             std::endl;
00424         std::cerr << memory_allocation_exception.what() << std::endl;
00425     }
00426     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00427
00428     #ifdef MTK_PRECISION_DOUBLE
00429     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00430     #else
00431     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00432     #endif
00433
00434     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00435         pp[ii] = pp[ii - 1] + mtk::kOne;
00436     }
00437
00438     #if MTK_DEBUG_LEVEL > 0
00439     std::cout << "pp =" << std::endl;
00440     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00441         std::cout << std::setw(12) << pp[ii];
00442     }
00443     std::cout << std::endl << std::endl;
00444     #endif
00445

```



```

00447
00448     bool transpose{false};
00449
00450     mtk::DenseMatrix vander_matrix(pp,
00451                                     order_accuracy_,
00452                                     order_accuracy_,
00453                                     transpose);
00454
00455     #if MTK_DEBUG_LEVEL > 0
00456     std::cout << "vander_matrix = " << std::endl;
00457     std::cout << vander_matrix << std::endl;
00458     #endif
00459
00460     try {
00461         coeffs_interior_ = new mtk::Real[order_accuracy_];
00462     } catch (std::bad_alloc &memory_allocation_exception) {
00463         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00464             std::endl;
00465         std::cerr << memory_allocation_exception.what() << std::endl;
00466     }
00467     memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00468
00469     coeffs_interior_[1] = mtk::kOne;
00470
00471     #if MTK_DEBUG_LEVEL > 0
00472     std::cout << "oo =" << std::endl;
00473     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00474         std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00475     }
00476     std::cout << std::endl;
00477     #endif
00478
00479     int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00480                                                  coeffs_interior_)};
00481
00482     #if MTK_DEBUG_LEVEL > 0
00483     if (!info) {
00484         std::cout << "System solved! Interior stencil attained!" << std::endl;
00485         std::cout << std::endl;
00486     }
00487     else {
00488         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00489         std::cerr << "Exiting..." << std::endl;
00490         return false;
00491     }
00492     #endif
00493
00494     #if MTK_DEBUG_LEVEL > 0
00495     std::cout << "coeffs_interior_ =" << std::endl;
00496     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00497         std::cout << std::setw(12) << coeffs_interior_[ii];
00498     }
00499     std::cout << std::endl << std::endl;
00500     #endif
00501
00502     delete [] pp;
00503     pp = nullptr;
00504
00505     return true;
00506 }
00507
00508 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
00509     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00510
00511     try {
00512         gg = new mtk::Real[num_bndy_coeffs_];
00513     } catch (std::bad_alloc &memory_allocation_exception) {
00514         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00515             std::endl;
00516         std::cerr << memory_allocation_exception.what() << std::endl;
00517     }
00518     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00519
00520     #ifdef MTK_PRECISION_DOUBLE
00521     gg[0] = -1.0/2.0;
00522     #else
00523     gg[0] = -1.0f/2.0f;
00524     #endif

```

```

00531     #endif
00532     for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00533         gg[ii] = gg[ii - 1] + mtk::kOne;
00534     }
00535
00536     #if MTK_DEBUG_LEVEL > 0
00537     std::cout << "gg =" << std::endl;
00538     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00539         std::cout << std::setw(12) << gg[ii];
00540     }
00541     std::cout << std::endl << std::endl;
00542     #endif
00543
00544     bool tran{true}; // Should I transpose the Vandermonde matrix.
00545
00546     mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00547
00548     #if MTK_DEBUG_LEVEL > 0
00549     std::cout << "vv_west_t =" << std::endl;
00550     std::cout << vv_west_t << std::endl;
00551     #endif
00552
00553     mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00554     (vv_west_t));
00555
00556     #if MTK_DEBUG_LEVEL > 0
00557     std::cout << "QQ^T =" << std::endl;
00558     std::cout << qq_t << std::endl;
00559     #endif
00560
00561     int KK_num_rows_{num_bndy_coeffs_};
00562     int KK_num_cols_{dim_null_};
00563
00564     mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00565
00566     for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00567         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00568             KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00569                 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00570         }
00571     }
00572
00573     #if MTK_DEBUG_LEVEL > 0
00574     std::cout << "KK =" << std::endl;
00575     std::cout << KK << std::endl;
00576     std::cout << "KK.num_rows() =" << KK.num_rows() << std::endl;
00577     std::cout << "KK.num_cols() =" << KK.num_cols() << std::endl;
00578     std::cout << std::endl;
00579     #endif
00580
00581     // Scale thus requesting that the last entries of the attained basis for the
00582     // null-space, adopt the pattern we require.
00583     // Essentially we will implement the following MATLAB pseudo-code:
00584     // scalers = KK(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00585     // SK = KK*scalers
00586     // where SK is the scaled null-space.
00587
00588     // In this point, we almost have all the data we need correctly allocated
00589     // in memory. We will create the matrix II_, and elements we wish to scale in
00590     // the KK array. Using the concept of the leading dimension, we could just
00591     // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00592     // GET how does it work. So I will just create a matrix with the content of
00593     // this array that we need, solve for the scalers and then scale the
00594     // whole KK:
00595
00596     // We will then create memory for that sub-matrix of KK (SUBK).
00597
00598     mtk::DenseMatrix SUBK(dim_null_, dim_null_);
00599
00600     for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00601         for (auto jj = 0; jj < dim_null_; ++jj) {
00602             SUBK.data()[ii - (num_bndy_coeffs_ - dim_null_)*dim_null_ + jj] =
00603                 KK.data()[ii*dim_null_ + jj];
00604         }
00605     }
00606
00607     #if MTK_DEBUG_LEVEL > 0

```

```

00615     std::cout << "SUBK =" << std::endl;
00616     std::cout << SUBK << std::endl;
00617     #endif
00618
00619     SUBK.Transpose();
00620
00621     #if MTK_DEBUG_LEVEL > 0
00622     std::cout << "SUBK^T =" << std::endl;
00623     std::cout << SUBK << std::endl;
00624     #endif
00625
00626     bool padded{false};
00627     tran = false;
00628
00629     mtk::DenseMatrix II(dim_null_, padded, tran);
00630
00631     #if MTK_DEBUG_LEVEL > 0
00632     std::cout << "II =" << std::endl;
00633     std::cout << II << std::endl;
00634     #endif
00635
00636     // Solve the system to compute the scalars.
00637     // An example of the system to solve, for k = 8, is:
00638     //
00639     // SUBK*scalars = II_ or
00640     //
00641     // | 0.386018  -0.0339244  -0.129478 |           | 1 0 0 |
00642     // | -0.119774  0.0199423  0.0558632 |*scalars = | 0 1 0 |
00643     // | 0.0155708 -0.00349546 -0.00853182 |           | 0 0 1 |
00644     //
00645     // Notice this is a nrhs = 3 system.
00646     // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalars... they
00647     // will be stored in the created identity matrix.
00648     // Let us first transpose SUBK (because of LAPACK):
00649
00650     int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00651
00652     #if MTK_DEBUG_LEVEL > 0
00653     if (!info) {
00654         std::cout << "System successfully solved!" <<
00655             std::endl;
00656     } else {
00657         std::cerr << "Something went wrong solving system! info = " << info <<
00658             std::endl;
00659         std::cerr << "Exiting..." << std::endl;
00660         return false;
00661     }
00662     std::cout << std::endl;
00663     #endif
00664
00665     #if MTK_DEBUG_LEVEL > 0
00666     std::cout << "Computed scalars:" << std::endl;
00667     std::cout << II << std::endl;
00668     #endif
00669
00670     // Multiply the two matrices to attain a scaled basis for null-space.
00671
00672     rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00673
00674     #if MTK_DEBUG_LEVEL > 0
00675     std::cout << "Rational basis for the null-space:" << std::endl;
00676     std::cout << rat_basis_null_space_ << std::endl;
00677     #endif
00678
00679     // At this point, we have a rational basis for the null-space, with the
00680     // pattern we need! :)
00681
00682     delete [] gg;
00683     gg = nullptr;
00684
00685     return true;
00686 }
00687
00688 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00689
00690
00691     mtk::Real *gg{}; // Generator vector for the first approximation.
00692
00693     try {
00694         gg = new mtk::Real[num_bndy_coeffs_];
00695     } catch (std::bad_alloc &memory_allocation_exception) {

```

```

00697     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00698     std::endl;
00699     std::cerr << memory_allocation_exception.what() << std::endl;
00700 }
00701 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00702
00703 #ifdef MTK_PRECISION_DOUBLE
00704 gg[0] = -1.0/2.0;
00705 #else
00706 gg[0] = -1.0f/2.0f;
00707 #endif
00708 for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00709     gg[ii] = gg[ii - 1] + mtk::kOne;
00710 }
00711
00712 #if MTK_DEBUG_LEVEL > 0
00713 std::cout << "gg0 =" << std::endl;
00714 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00715     std::cout << std::setw(12) << gg[ii];
00716 }
00717 std::cout << std::endl << std::endl;
00718 #endif
00719
00720 // Allocate 2D array to store the collection of preliminary approximations.
00721 try {
00722     prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00723 } catch (std::bad_alloc &memory_allocation_exception) {
00724     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00725     std::endl;
00726     std::cerr << memory_allocation_exception.what() << std::endl;
00727 }
00728 memset(prem_apps_,
00729         mtk::kZero,
00730         sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00731
00732
00733 for (auto ll = 0; ll < dim_null_; ++ll) {
00734
00735     // Re-check new generator vector for every iteration except for the first.
00736     #if MTK_DEBUG_LEVEL > 0
00737     if (ll > 0) {
00738         std::cout << "gg" << ll << " =" << std::endl;
00739         for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00740             std::cout << std::setw(12) << gg[ii];
00741         }
00742         std::cout << std::endl << std::endl;
00743     }
00744     #endif
00745
00746     bool transpose{false};
00747
00748     mtk::DenseMatrix AA_(gg,
00749                           num_bndy_coeffs_, order_accuracy_ + 1,
00750                           transpose);
00751
00752     #if MTK_DEBUG_LEVEL > 0
00753     std::cout << "AA_" << ll << " =" << std::endl;
00754     std::cout << AA_ << std::endl;
00755     #endif
00756
00757     mtk::Real *ob{};
00758
00759     auto ob_ld = num_bndy_coeffs_;
00760
00761     try {
00762         ob = new mtk::Real[ob_ld];
00763     } catch (std::bad_alloc &memory_allocation_exception) {
00764         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00765         std::endl;
00766         std::cerr << memory_allocation_exception.what() << std::endl;
00767     }
00768     memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00769
00770     ob[1] = mtk::kOne;
00771
00772     #if MTK_DEBUG_LEVEL > 0
00773     std::cout << "ob =" << std::endl << std::endl;
00774     for (auto ii = 0; ii < ob_ld; ++ii) {
00775         std::cout << std::setw(12) << ob[ii] << std::endl;
00776     }
00777

```

```

00781     }
00782     std::cout << std::endl;
00783     #endif
00784
00785
00786
00787     // However, this is an under-determined system of equations. So we can not
00788     // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00789     // our LAPACKAdapter class.
00790
00791     int info_{
00792         mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00793             ob, ob_ld);
00794
00795     #if MTK_DEBUG_LEVEL > 0
00796     if (!info_) {
00797         std::cout << "System successfully solved!" << std::endl << std::endl;
00798     } else {
00799         std::cerr << "Error solving system! info = " << info_ << std::endl;
00800     }
00801     #endif
00802
00803     #if MTK_DEBUG_LEVEL > 0
00804     std::cout << "ob =" << std::endl;
00805     for (auto ii = 0; ii < ob_ld; ++ii) {
00806         std::cout << std::setw(12) << ob[ii] << std::endl;
00807     }
00808     std::cout << std::endl;
00809     #endif
00810
00811
00812     // This implies a DAXPY operation. However, we must construct the arguments
00813     // for this operation.
00814
00815     // Save them into the ob_bottom array:
00816
00817     Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00818
00819     try {
00820         ob_bottom = new mtk::Real[dim_null_];
00821     } catch (std::bad_alloc &memory_allocation_exception) {
00822         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00823             std::endl;
00824         std::cerr << memory_allocation_exception.what() << std::endl;
00825     }
00826     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00827
00828     for (auto ii = 0; ii < dim_null_; ++ii) {
00829         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00830     }
00831
00832     #if MTK_DEBUG_LEVEL > 0
00833     std::cout << "ob_bottom =" << std::endl;
00834     for (auto ii = 0; ii < dim_null_; ++ii) {
00835         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00836     }
00837     std::cout << std::endl;
00838     #endif
00839
00840
00841
00842
00843     // We must computed an scaled ob, sob, using the scaled null-space in
00844     // rat_basis_null_space_.
00845     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00846     // or:                      ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00847     // thus:                    Y =      a*A      *x      +      b*Y (DAXPY).
00848
00849     #if MTK_DEBUG_LEVEL > 0
00850     std::cout << "Rational basis for the null-space:" << std::endl;
00851     std::cout << rat_basis_null_space_ << std::endl;
00852     #endif
00853
00854     mtk::Real alpha{-mtk::kOne};
00855     mtk::Real beta{mtk::kOne};
00856
00857     mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00858         ob_bottom, beta, ob);
00859
00860     #if MTK_DEBUG_LEVEL > 0
00861     std::cout << "scaled ob:" << std::endl;
00862     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00863         std::cout << std::setw(12) << ob[ii] << std::endl;
00864     }

```

```

00865     std::cout << std::endl;
00866     #endif
00867
00868     // We save the recently scaled solution, into an array containing these.
00869     // We can NOT start building the pi matrix, simply because I want that part
00870     // to be separated since its construction depends on the algorithm we want
00871     // to implement.
00872
00873     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00874         prem_apps_[ii*dim_null_ + ll] = ob[ii];
00875     }
00876
00877     // After the first iteration, simply shift the entries of the last
00878     // generator vector used:
00879     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00880         gg[ii]--;
00881     }
00882
00883     // Garbage collection for this loop:
00884     delete[] ob;
00885     ob = nullptr;
00886
00887     delete[] ob_bottom;
00888     ob_bottom = nullptr;
00889 } // End of: for (ll = 0; ll < dim_null; ll++);
00890
00891 #if MTK_DEBUG_LEVEL > 0
00892 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00893 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00894     for (auto jj = 0; jj < dim_null_; ++jj) {
00895         std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00896     }
00897     std::cout << std::endl;
00898 }
00899 std::cout << std::endl;
00900 #endif
00901
00902 delete[] gg;
00903 gg = nullptr;
00904
00905 return true;
00906 }
00907
00908 bool mtk::Div1D::ComputeWeights(void) {
00909
00910     // Matrix to compute the weights as in the CRSA.
00911     mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00912
00913
00914     // Assemble the pi matrix using:
00915     // 1. The collection of scaled preliminary approximations.
00916     // 2. The collection of coefficients approximating at the interior.
00917     // 3. The scaled basis for the null-space.
00918
00919
00920     // 1.1. Process array of scaled preliminary approximations.
00921
00922     // These are queued in scaled_solutions. Each one of these, will be a column
00923     // of the pi matrix:
00924     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00925         for (auto jj = 0; jj < dim_null_; ++jj) {
00926             pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00927                 prem_apps_[ii*dim_null_ + jj];
00928         }
00929     }
00930
00931     // 1.2. Add columns from known stencil approximating at the interior.
00932
00933     // However, these must be padded by zeros, according to their position in the
00934     // final pi matrix:
00935     auto mm = 0;
00936     for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
00937         for (auto ii = 0; ii < order_accuracy_; ++ii) {
00938             pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00939                 coeffs_interior_[ii];
00940         }
00941         ++mm;
00942     }
00943
00944     rat_basis_null_space_.OrderColMajor();
00945
00946     #if MTK_DEBUG_LEVEL > 0

```

```

00947     std::cout << "Rational basis for the null-space (col. major):" << std::endl;
00948     std::cout << rat_basis_null_space_ << std::endl;
00949     #endif
00950
00951     // 1.3. Add final set of columns: rational basis for null-space.
00952     for (auto jj = dim_null_ + (order_accuracy_/2 + 1); jj < num_bndy_coeffs_ - 1; ++jj) {
00953         for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00954             auto og =
00955                 (jj - (dim_null_ + (order_accuracy_/2 + 1))) * num_bndy_coeffs_ + ii;
00956             auto de = ii * (2 * dim_null_ + (order_accuracy_/2 + 1)) + jj;
00957             pi.data()[de] = rat_basis_null_space_.data()[og];
00958         }
00959     }
00960
00961     #if MTK_DEBUG_LEVEL > 0
00962     std::cout << "coeffs_interior_" << std::endl;
00963     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00964         std::cout << std::setw(12) << coeffs_interior_[ii];
00965     }
00966     std::cout << std::endl << std::endl;
00967     #endif
00968
00969     #if MTK_DEBUG_LEVEL > 0
00970     std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
00971     std::cout << pi << std::endl;
00972     #endif
00973
00974     // This imposes the mimetic condition.
00975
00976     mtk::Real *hh{}; // Right-hand side to compute weights in the C(R,B)SA.
00977
00978     try {
00979         hh = new mtk::Real[num_bndy_coeffs_];
00980     } catch (std::bad_alloc &memory_allocation_exception) {
00981         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00982             std::endl;
00983         std::cerr << memory_allocation_exception.what() << std::endl;
00984     }
00985     memset(hh, mtk::kZero, sizeof(hh[0]) * num_bndy_coeffs_);
00986
00987     hh[0] = -mtk::kOne;
00988     for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
00989         auto aux_xx = mtk::kZero;
00990         for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
00991             aux_xx += coeffs_interior_[jj];
00992         }
00993         hh[ii] = -mtk::kOne * aux_xx;
00994     }
00995
00996     // That is, we construct a system, to solve for the weights.
00997
00998     // Once again we face the challenge of solving with LAPACK. However, for the
00999     // CRSA, this matrix PI is over-determined, since it has more rows than
01000     // unknowns. However, according to the theory, the solution to this system is
01001     // unique. We will use dgels_.
01002
01003     try {
01004         weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01005     } catch (std::bad_alloc &memory_allocation_exception) {
01006         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01007             std::endl;
01008         std::cerr << memory_allocation_exception.what() << std::endl;
01009     }
01010     memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0]) * num_bndy_coeffs_);
01011
01012     int weights_ld{pi.num_cols() + 1};
01013
01014     // Preserve hh.
01015     std::copy(hh, hh + weights_ld, weights_cbs_);
01016
01017     pi.Transpose();
01018
01019     int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
01020         pi, weights_cbs_, weights_ld)};
01021
01022     #if MTK_DEBUG_LEVEL > 0
01023     if (!info) {
01024         std::cout << "System successfully solved!" << std::endl << std::endl;
01025     } else {

```

```

01029     std::cerr << "Error solving system! info = " << info << std::endl;
01030 }
01031 #endif
01032
01033 #if MTK_DEBUG_LEVEL > 0
01034 std::cout << "hh =" << std::endl;
01035 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01036     std::cout << std::setw(11) << hh[ii] << std::endl;
01037 }
01038 std::cout << std::endl;
01039 #endif
01040
01041 // Preserve the original weights for research.
01042
01043 try {
01044     weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01045 } catch (std::bad_alloc &memory_allocation_exception) {
01046     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01047         std::endl;
01048     std::cerr << memory_allocation_exception.what() << std::endl;
01049 }
01050 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01051
01052 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01053
01054 #if MTK_DEBUG_LEVEL > 0
01055 std::cout << "weights_CRSA + lambda =" << std::endl;
01056 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01057     std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01058 }
01059 std::cout << std::endl;
01060 #endif
01061
01062 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01063     int minrow_{std::numeric_limits<int>::infinity()};
01064
01065     mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01066         order_accuracy_)};
01067     mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01068
01069     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01070
01071     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01072         for (auto jj = 0; jj < dim_null_; ++jj) {
01073             phi.data()[ii*(order_accuracy_ + 1) + jj] = prem_apps_[ii*dim_null_ + jj];
01074         }
01075     }
01076
01077     int aux{}; // Auxiliary variable.
01078     for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01079         for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01080             phi.data()[ii*(order_accuracy_ + 1) + jj] = coeffs_interior[ii];
01081         }
01082         ++aux;
01083     }
01084
01085     for (auto jj = order_accuracy_ - 1; jj >= order_accuracy_ - dim_null_; jj--) {
01086         for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01087             phi.data()[ii*(order_accuracy_ + 1) + jj] = mtk::kZero;
01088         }
01089     }
01090
01091     for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01092         for (auto ii = 0; ii < dim_null_; ++ii) {
01093             phi.data()[ii*(order_accuracy_ + 1) + jj] =
01094                 -pre_apps_[(dim_null_ - ii - 1 + jj*dim_null_)];
01095         }
01096     }
01097
01098     for (auto ii = 0; ii < order_accuracy_/2; ++ii) {
01099         for (auto jj = dim_null_ + 2; jj < order_accuracy_ + 1; ++jj) {
01100             auto swap = phi.data()[ii*(order_accuracy_ + 1) + jj];
01101             phi.data()[ii*(order_accuracy_ + 1) + jj] =
01102                 phi.data()[ii*(order_accuracy_ + 1) + jj + dim_null_];
01103             phi.data()[ii*(order_accuracy_ + 1) + jj + dim_null_] = swap;
01104         }
01105     }
01106 }
01107
01108 }
01109
01110

```



```

01111     #if MTK_DEBUG_LEVEL > 0
01112     std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01113     std::cout << phi << std::endl;
01114     #endif
01115
01117     mtk::Real *lamed{}; // Used to build big lambda.
01118
01119     try {
01120         lamed = new mtk::Real[dim_null_];
01121     } catch (std::bad_alloc &memory_allocation_exception) {
01122         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01123             std::endl;
01124         std::cerr << memory_allocation_exception.what() << std::endl;
01125     }
01126     memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01127
01128     for (auto ii = 0; ii < dim_null_; ++ii) {
01129         lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01130     }
01131
01132     #if MTK_DEBUG_LEVEL > 0
01133     std::cout << "lamed =" << std::endl;
01134     for (auto ii = 0; ii < dim_null_; ++ii) {
01135         std::cout << std::setw(12) << lamed[ii] << std::endl;
01136     }
01137     std::cout << std::endl;
01138     #endif
01139
01140     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01141         mtk::Real temp = mtk::kZero;
01142         for (auto jj = 0; jj < dim_null_; ++jj) {
01143             temp = temp +
01144                 lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01145         }
01146         hh[ii] = hh[ii] - temp;
01147     }
01148
01149     #if MTK_DEBUG_LEVEL > 0
01150     std::cout << "big_lambda =" << std::endl;
01151     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01152         std::cout << std::setw(12) << hh[ii] << std::endl;
01153     }
01154     std::cout << std::endl;
01155     #endif
01156
01157     int copy_result{};
01158
01159     mtk::Real normerr_; // Norm of the error for the solution on each row.
01160
01161     for (auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01162         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01163             data(),
01164                 order_accuracy_ + 1,
01165                 order_accuracy_,
01166                 order_accuracy_,
01167                 hh,
01168                 weights_cbs_,
01169                 row_,
01170                 mimetic_threshold_,
01171                 copy_result);
01172
01173         mtk::Real aux{normerr_/norm_};
01174
01175         #if MTK_DEBUG_LEVEL>0
01176         std::cout << "Relative norm: " << aux << " " << std::endl;
01177         std::cout << std::endl;
01178         #endif
01179
01180         if (aux < minnorm_) {
01181             minnorm_ = aux;
01182             minrow_ = row_;
01183         }
01184     }
01185
01186     #if MTK_DEBUG_LEVEL > 0
01187     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01188         std::endl;
01189     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01190         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01191     }
01192

```

```

01193     std::cout << std::endl;
01194     #endif
01195
01196
01197
01198     // After we know which row yields the smallest relative norm that row is
01199     // chosen to be the objective function and the result of the optimizer is
01200     // chosen to be the new weights_.
01201
01202     #if MTK_DEBUG_LEVEL > 0
01203     std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01204         minrow_ + 1 << std::endl;
01205     std::cout << std::endl;
01206     #endif
01207
01208     copy_result = 1;
01209     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01210
01211         order_accuracy_ + 1,
01212         order_accuracy_,
01213         order_accuracy_,
01214         hh,
01215         weights_cbs_,
01216         minrow_,
01217         mimetic_threshold_,
01218         copy_result);
01219     mtk::Real aux_{normerr_/norm_};
01220     #if MTK_DEBUG_LEVEL > 0
01221     std::cout << "Relative norm: " << aux_ << std::endl;
01222     std::cout << std::endl;
01223     #endif
01224     delete [] lamed;
01225     lamed = nullptr;
01226 }
01227
01228 delete [] hh;
01229 hh = nullptr;
01230
01231 return true;
01232 }
01233
01234 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01235
01236     #if MTK_DEBUG_LEVEL > 0
01237     std::cout << "weights_CBSA + lambda =" << std::endl;
01238     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01239         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01240     }
01241     std::cout << std::endl;
01242     #endif
01243
01244
01245
01246     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01247
01248     try {
01249         lambda = new mtk::Real[dim_null_];
01250     } catch (std::bad_alloc &memory_allocation_exception) {
01251         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01252             std::endl;
01253         std::cerr << memory_allocation_exception.what() << std::endl;
01254     }
01255     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01256
01257     for (auto ii = 0; ii < dim_null_; ++ii) {
01258         lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01259     }
01260
01261     #if MTK_DEBUG_LEVEL > 0
01262     std::cout << "lambda =" << std::endl;
01263     for (auto ii = 0; ii < dim_null_; ++ii) {
01264         std::cout << std::setw(12) << lambda[ii] << std::endl;
01265     }
01266     std::cout << std::endl;
01267     #endif
01268
01269
01270
01271     mtk::Real *alpha{}; // Collection of alpha values.
01272
01273     try {
01274         alpha = new mtk::Real[dim_null_];
01275     } catch (std::bad_alloc &memory_allocation_exception) {

```

```

01276     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01277     std::endl;
01278     std::cerr << memory_allocation_exception.what() << std::endl;
01279 }
01280 memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01281
01282 for (auto ii = 0; ii < dim_null_; ++ii) {
01283     alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01284 }
01285
01286 #if MTK_DEBUG_LEVEL > 0
01287 std::cout << "alpha =" << std::endl;
01288 for (auto ii = 0; ii < dim_null_; ++ii) {
01289     std::cout << std::setw(12) << alpha[ii] << std::endl;
01290 }
01291 std::cout << std::endl;
01292 #endif
01293
01295
01296 try {
01297     mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01298 } catch (std::bad_alloc &memory_allocation_exception) {
01299     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01300     std::endl;
01301     std::cerr << memory_allocation_exception.what() << std::endl;
01302 }
01303 memset(mim_bndy_, mtk::kZero, sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01304
01305 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01306     for (auto jj = 0; jj < dim_null_; ++jj) {
01307         mim_bndy_[ii*dim_null_ + jj] =
01308             prem_apps_[ii*dim_null_ + jj] +
01309             alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01310     }
01311 }
01312
01313 #if MTK_DEBUG_LEVEL > 0
01314 std::cout << "Collection of mimetic approximations:" << std::endl;
01315 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01316     for (auto jj = 0; jj < dim_null_; ++jj) {
01317         std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01318     }
01319     std::cout << std::endl;
01320 }
01321 std::cout << std::endl;
01322 #endif
01323
01324 delete[] lambda;
01325 lambda = nullptr;
01326
01327 delete[] alpha;
01328 alpha = nullptr;
01329
01330 return true;
01331 }
01332
01333 bool mtk::Div1D::AssembleOperator(void) {
01334
01335     // The output array will have this form:
01336     // 1. The first entry of the array will contain the used order order_accuracy_.
01337     // 2. The second entry of the array will contain the collection of
01338     // approximating coefficients for the interior of the grid.
01339     // 3. IF order_accuracy_ > 2, then the third entry will contain a collection of weights.
01340     // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the collections of
01341     // approximating coefficients for the west boundary of the grid.
01342
01343     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01344         divergence_length_ =
01345             1 + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01346     } else {
01347         divergence_length_ = 1 + order_accuracy_;
01348     }
01349
01350 #if MTK_DEBUG_LEVEL > 0
01351 std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01352 #endif
01353
01354 try {
01355     divergence_ = new double[divergence_length_];
01356 } catch (std::bad_alloc &memory_allocation_exception) {
01357     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```

```

01358         std::endl;
01359         std::cerr << memory_allocation_exception.what() << std::endl;
01360     }
01361     memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01362
01363
01364
01365     divergence_[0] = order_accuracy_;
01366
01367
01368
01369     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01370         divergence_[ii + 1] = coeffs_interior_[ii];
01371     }
01372
01373
01374
01375     if (order_accuracy_ > 2) {
01376         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01377             divergence_[1 + order_accuracy_ + ii] = weights_cbs_[ii];
01378         }
01379     }
01380
01381
01382
01383
01384     if (order_accuracy_ > 2) {
01385         auto offset = (2*order_accuracy_ + 1);
01386         int mm{};
01387         for (auto ii = 0; ii < dim_null_; ++ii) {
01388             for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01389                 divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01390                 ++mm;
01391             }
01392         }
01393     }
01394
01395     #if MTK_DEBUG_LEVEL > 0
01396     std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01397     std::cout << std::endl;
01398     #endif
01399
01400     return true;
01401 }

```

17.63 src/mtk_div_2d.cc File Reference

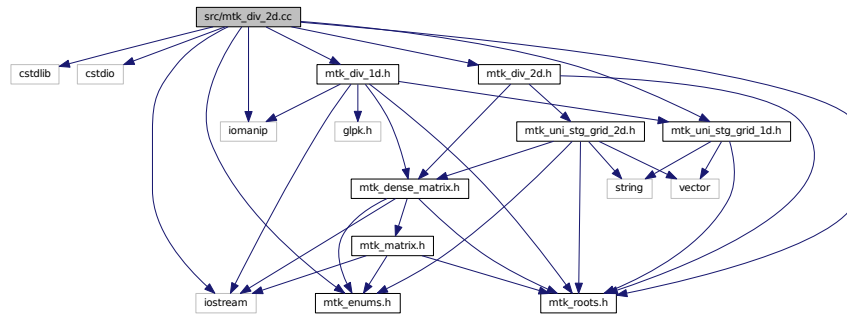
Implements the class Div2D.

```

#include <cstdlib>
#include <stdio>
#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:



17.63.1 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](#).

17.64 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; // Gx vertical dimension
00087     int nx = num_cells_x + 1; // Gx horizontal dimension
00088     int my = num_cells_y + 2; // Gy vertical dimension
00089     int ny = num_cells_y + 1; // Gy horizontal dimension
00090
00091     mtk::Div1D div;
00092
00093     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00094
00095     if (!info) {
00096         std::cerr << "Mimetic div could not be built." << std::endl;
00097         return info;
00098     }
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 dx(div.ReturnAsDenseMatrix(grid_x));
00109     mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00110
00111     bool padded{true};
00112     bool transpose{false};
00113
00114     mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00115     mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00116
00117     mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00118     mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00119
00120     #if MTK_DEBUG_LEVEL > 0
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 << "Kronecker dimensions Grad 2D" <<
00126     mx*num_cells_y + my*num_cells_x << " by " << nx*ny << std::endl;
00127     #endif
00128
00129     mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00130
00131     for (auto ii = 0; ii < mx*my; ii++) {
00132         for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00133             d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00134         }
00135         for (auto kk=0; kk<ny*num_cells_x; kk++) {
00136             d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00137         }
00138     }
00139
00140     divergence_ = d2d;
00141
00142     return info;
00143 }
00144
00145 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() {
00146
00147     return divergence_;
00148 }

```

17.65 src/mtk_glpk_adapter.cc File Reference

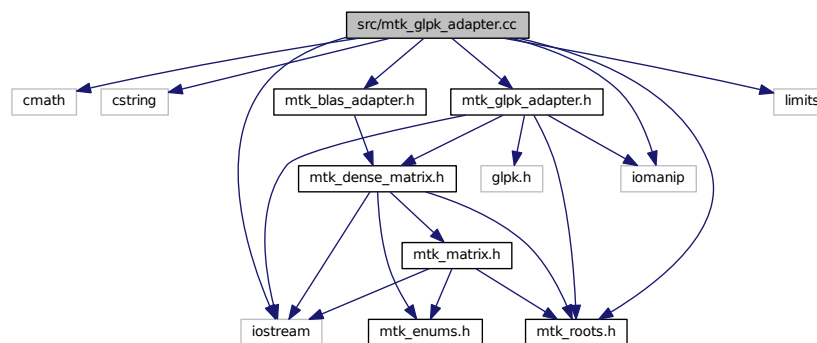
Adapter class for the GLPK API.

```

#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:



17.65.1 Detailed Description

This class contains a collection of static classes, 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](#).

17.66 mtk_glpk_adapter.cc

```
00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
00026 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00027 and a copy of the modified files should be reported once modifications are
00028 completed, unless these modifications are made through the project's GitHub
00029 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00030 should be developed and included in any deliverable.
00031
00032 2. Redistributions of source code must be done through direct
00033 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00034
00035 3. Redistributions in binary form must reproduce the above copyright notice,
00036 this list of conditions and the following disclaimer in the documentation and/or
00037 other materials provided with the distribution.
00038
00039 4. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders, and due credit should
00041 be given to the copyright holders.
00042
00043 5. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does
00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #include <cmath>
00066 #include <cstring>
00067
00068 #include <iostream>
00069 #include <iomanip>
00070 #include <limits>
00071
00072 #include "mtk_roots.h"
```



```

00073 #include "mtk_blas_adapter.h"
00074 #include "mtk_glpk_adapter.h"
00075
00076 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
    mtk::Real *A,
00077                                     int nrows,
00078                                     int ncols,
00079                                     int kk,
00080                                     mtk::Real *hh,
00081                                     mtk::Real *qq,
00082                                     int robjective,
00083                                     mtk::Real mimetic_threshold,
00084                                     int copy) {
00085
00086     #if MTK_DEBUG_LEVEL > 0
00087     char mps_file_name[18]; // File name for the MPS files.
00088     #endif
00089     char rname[5];          // Row name.
00090     char cname[5];          // Column name.
00091
00092     glp_prob *lp; // Linear programming problem.
00093
00094     int *ia; // Array for the problem.
00095     int *ja; // Array for the problem.
00096
00097     int problem_size; // Size of the problem.
00098     int lp_nrows;     // Number of rows.
00099     int lp_ncols;     // Number of columns.
00100     int matsize;      // Size of the matrix.
00101     int glp_index{1}; // Index of the objective function.
00102     int ii;           // Iterator.
00103     int jj;           // Iterator.
00104
00105     mtk::Real *ar; // Array for the problem.
00106     mtk::Real *objective; // Array containing the objective function.
00107     mtk::Real *rhs; // Array containing the rhs.
00108     mtk::Real *err; // Array of errors.
00109
00110     mtk::Real x1; // Norm-2 of the error.
00111
00112     #if MTK_DEBUG_LEVEL > 0
00113     mtk::Real obj_value; // Value of the objective function.
00114     #endif
00115
00116     lp_nrows = kk;
00117     lp_ncols = kk;
00118
00119     matsize = lp_nrows*lp_ncols;
00120
00121
00122
00123
00124     problem_size = lp_nrows*lp_ncols + 1;
00125
00126     try {
00127         ia = new int[problem_size];
00128     } catch (std::bad_alloc &memory_allocation_exception) {
00129         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00130             std::endl;
00131         std::cerr << memory_allocation_exception.what() << std::endl;
00132     }
00133     memset(ia, 0, sizeof(ia[0])*problem_size);
00134
00135     try {
00136         ja = new int[problem_size];
00137     } catch (std::bad_alloc &memory_allocation_exception) {
00138         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00139             std::endl;
00140         std::cerr << memory_allocation_exception.what() << std::endl;
00141     }
00142     memset(ja, 0, sizeof(ja[0])*problem_size);
00143
00144     try {
00145         ar = new mtk::Real[problem_size];
00146     } catch (std::bad_alloc &memory_allocation_exception) {
00147         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00148             std::endl;
00149         std::cerr << memory_allocation_exception.what() << std::endl;
00150     }
00151     memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00152
00153     try {
00154         objective = new mtk::Real[lp_ncols + 1];

```

```

00155 } catch (std::bad_alloc &memory_allocation_exception) {
00156     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00157         std::endl;
00158     std::cerr << memory_allocation_exception.what() << std::endl;
00159 }
00160 memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00161
00162 try {
00163     rhs = new mtk::Real[lp_nrows + 1];
00164 } catch (std::bad_alloc &memory_allocation_exception) {
00165     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00166         std::endl;
00167     std::cerr << memory_allocation_exception.what() << std::endl;
00168 }
00169 memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00170
00171 try {
00172     err = new mtk::Real[lp_nrows];
00173 } catch (std::bad_alloc &memory_allocation_exception) {
00174     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00175         std::endl;
00176     std::cerr << memory_allocation_exception.what() << std::endl;
00177 }
00178 memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00179
00180 #if MTK_DEBUG_LEVEL > 0
00181 std::cout << "Problem size: " << problem_size << std::endl;
00182 std::cout << "lp_nrows = " << lp_nrows << std::endl;
00183 std::cout << "lp_ncols = " << lp_ncols << std::endl;
00184 std::cout << std::endl;
00185 #endif
00186
00187 lp = glp_create_prob();
00188
00189 glp_set_prob_name (lp, "mtk:GLPKAdapter::Simplex");
00190
00191 glp_set_obj_dir (lp, GLP_MIN);
00192
00193
00194
00195 glp_add_rows(lp, lp_nrows);
00196
00197 for (ii = 1; ii <= lp_nrows; ++ii) {
00198     sprintf(rname, "R%02d",ii);
00199     glp_set_row_name(lp, ii, rname);
00200 }
00201
00202 glp_add_cols(lp, lp_ncols);
00203
00204 for (ii = 1; ii <= lp_ncols; ++ii) {
00205     sprintf(cname, "Q%02d",ii);
00206     glp_set_col_name (lp, ii, cname);
00207 }
00208
00209
00210
00211 #if MTK_DEBUG_LEVEL>0
00212 std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
00213 #endif
00214 for (jj = 0; jj < kk; ++jj) {
00215     objective[glp_index] = A[jj + robjective * ncols];
00216     glp_index++;
00217 }
00218 #if MTK_DEBUG_LEVEL > 0
00219 std::cout << std::endl;
00220 #endif
00221
00222
00223
00224 glp_index = 1;
00225 rhs[0] = mtk::kZero;
00226 for (ii = 0; ii <= lp_nrows; ++ii) {
00227     if (ii != robjective) {
00228         rhs[glp_index] = hh[ii];
00229         glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00230         glp_index++;
00231     }
00232 }
00233
00234 #if MTK_DEBUG_LEVEL > 0
00235 std::cout << "rhs =" << std::endl;
00236 for (auto ii = 0; ii < lp_nrows; ++ii) {
00237     std::cout << std::setw(15) << rhs[ii] << std::endl;
00238 }

```

```

00239     std::cout << std::endl;
00240     #endif
00241
00242
00243
00244     for (ii = 1; ii <= lp_ncols; ++ii) {
00245         glp_set_obj_coef (lp, ii, objective[ii]);
00246     }
00247
00248
00249
00250     for (ii = 1; ii <= lp_ncols; ++ii) {
00251         glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00252     }
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     glp_index = 0;
00267
00268     for (ii = 1; ii < problem_size; ++ii) {
00269         if ((ii - 1) % lp_ncols == 0) {
00270             glp_index++;
00271         }
00272         ia[ii] = glp_index;
00273         ja[ii] = (ii - 1) % lp_ncols + 1;
00274     }
00275
00276     glp_load_matrix (lp, matsize, ia, ja, ar);
00277
00278     #if MTK_DEBUG_LEVEL > 0
00279     sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00280     glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00281     #endif
00282
00283
00284
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 }

```

17.67 src/mtk_grad_1d.cc File Reference

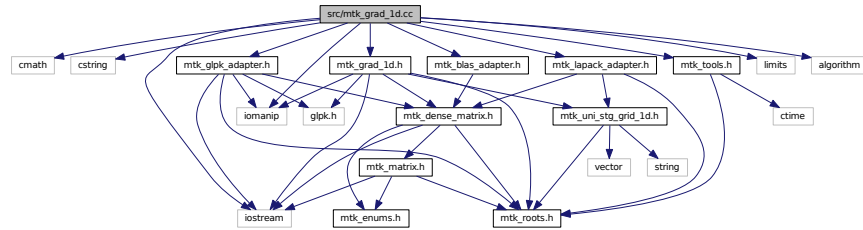
Implements the class Grad1D.

```

#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)`

17.67.1 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`.

17.68 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 #include <limits>
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074

```

```

00075 #include "mtk_grad_ld.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::GradLD &in) {
00080
00082     stream << "gradient_[0] = " << std::setw(9) << in.gradient_[0] << std::endl;
00083
00084
00086     stream << "gradient_[1:" << in.order_accuracy_ << "] = ";
00087     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00088         stream << std::setw(9) << in.gradient_[ii] << " ";
00089     }
00090     stream << std::endl;
00091
00092
00094     stream << "gradient_[\" << in.order_accuracy_ + 1 << ":" <<
00095         2*in.order_accuracy_ << "] = ";
00096     for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00097         order_accuracy_; ++ii) {
00098         stream << std::setw(9) << in.gradient_[ii] << " ";
00099     }
00100     stream << std::endl;
00101
00103     int offset{2*in.order_accuracy_ + 1};
00104     int mm {};
00105
00106     stream << "gradient_[\" << offset + mm << ":" <<
00107         offset + mm + in.num_bndy_coeffs_ - 1 << "] = ";
00108
00109     if (in.order_accuracy_ > mtk::kDefaultOrderAccuracy) {
00110         for (auto ii = 0; ii < in.num_bndy_approxs_ ; ++ii) {
00111             for (auto jj = 0; jj < in.num_bndy_coeffs_ ; ++jj) {
00112                 auto value = in.gradient_[offset + (mm)];
00113                 stream << std::setw(9) << value << " ";
00114                 mm++;
00115             }
00116         }
00117     } else {
00118         stream << std::setw(9) << in.gradient_[offset + 0] << ' ';
00119         stream << std::setw(9) << in.gradient_[offset + 1] << ' ';
00120         stream << std::setw(9) << in.gradient_[offset + 2] << ' ';
00121     }
00122     stream << std::endl;
00123
00124     return stream;
00125 }
00126 }
00127 }
00128
00129 mtk::GradLD::GradLD():
00130     order_accuracy_(mtk::kDefaultOrderAccuracy),
00131     dim_null_(),
00132     num_bndy_approxs_(),
00133     num_bndy_coeffs_(),
00134     gradient_length_(),
00135     minrow_(),
00136     row_(),
00137     coeffs_interior_(),
00138     prem_apps_(),
00139     weights_crs_(),
00140     weights_cbs_(),
00141     mim_bndy_(),
00142     gradient_(),
00143     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00144
00145 mtk::GradLD::GradLD(const GradLD &grad):
00146     order_accuracy_(grad.order_accuracy_),
00147     dim_null_(grad.dim_null_),
00148     num_bndy_approxs_(grad.num_bndy_approxs_),
00149     num_bndy_coeffs_(grad.num_bndy_coeffs_),
00150     gradient_length_(grad.gradient_length_),
00151     minrow_(grad.minrow_),
00152     row_(grad.row_),
00153     coeffs_interior_(grad.coeffs_interior_),
00154     prem_apps_(grad.prem_apps_),
00155     weights_crs_(grad.weights_crs_),
00156     weights_cbs_(grad.weights_cbs_),
00157     mim_bndy_(grad.mim_bndy_),
00158     gradient_(grad.gradient_),

```

```

00159   mimetic_threshold_(grad.mimetic_threshold_) {}
00160
00161 mtk::Grad1D::~Grad1D() {
00162
00163     delete[] coeffs_interior_;
00164     coeffs_interior_ = nullptr;
00165
00166     delete[] prem_apps_;
00167     prem_apps_ = nullptr;
00168
00169     delete[] weights_crs_;
00170     weights_crs_ = nullptr;
00171
00172     delete[] weights_cbs_;
00173     weights_cbs_ = nullptr;
00174
00175     delete[] mim_bndy_;
00176     mim_bndy_ = nullptr;
00177
00178     delete[] gradient_;
00179     gradient_ = nullptr;
00180 }
00181
00182 bool mtk::Grad1D::ConstructGrad1D(int order_accuracy,
00183   Real mimetic_threshold) {
00184
00185     #if MTK_DEBUG_LEVEL > 0
00186     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00187     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00188     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00189       __FILE__, __LINE__, __func__);
00189
00190     if (order_accuracy >= mtk::kCriticalOrderAccuracyGrad) {
00191         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00192     }
00193
00194     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00195     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00196     #endif
00197
00198     order_accuracy_ = order_accuracy;
00199     mimetic_threshold_ = mimetic_threshold;
00200
00202
00203     bool abort_construction = ComputeStencilInteriorGrid();
00204
00205     #if MTK_DEBUG_LEVEL > 0
00206     if (!abort_construction) {
00207         std::cerr << "Could NOT complete stage 1." << std::endl;
00208         std::cerr << "Exiting..." << std::endl;
00209         return false;
00210     }
00211     #endif
00212
00213     // At this point, we already have the values for the interior stencil stored
00214     // in the coeffs_interior_ array.
00215
00216     dim_null_ = order_accuracy_/2 - 1;
00217
00218     num_bndy_approx_ = dim_null_ + 1;
00219
00220     #ifdef MTK_PRECISION_DOUBLE
00221     num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00222     #else
00223     num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00224     #endif
00225
00227
00228     // For this we will follow recommendations given in:
00229     //
00230     // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00231     //
00232     // We will compute the QR Factorization of the transpose, as in the
00233     // following (MATLAB) pseudo-code:
00234     //
00235     // [Q,R] = qr(V'); % Full QR as defined in
00236     // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00237     //
00238     // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00239     //
00240     // However, given the nature of the Vandermonde matrices we've just

```

```

00241 // computed, they all posses the same null-space. Therefore, we impose the
00242 // convention of computing the null-space of the first Vandermonde matrix
00243 // (west boundary).
00244
00245 // In the case of the gradient, the first Vandermonde system has a unique
00246 // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00247 // matrix used to assemble said system, will have an empty null-space.
00248
00249 // Therefore, we only compute a rational basis for the case of order higher
00250 // than second.
00251
00252 if (dim_null_ > 0) {
00253     abort_construction = ComputeRationalBasisNullSpace();
00254
00255     #if MTK_DEBUG_LEVEL > 0
00256     if (!abort_construction) {
00257         std::cerr << "Could NOT complete stage 2.1." << std::endl;
00258         std::cerr << "Exiting..." << std::endl;
00259         return false;
00260     }
00261     #endif
00262 }
00263
00264
00266 abort_construction = ComputePreliminaryApproximations();
00267
00268 #if MTK_DEBUG_LEVEL > 0
00269 if (!abort_construction) {
00270     std::cerr << "Could NOT complete stage 2.2." << std::endl;
00271     std::cerr << "Exiting..." << std::endl;
00272     return false;
00273 }
00274 #endif
00275
00276 abort_construction = ComputeWeights();
00277
00278 #if MTK_DEBUG_LEVEL > 0
00279 if (!abort_construction) {
00280     std::cerr << "Could NOT complete stage 2.3." << std::endl;
00281     std::cerr << "Exiting..." << std::endl;
00282     return false;
00283 }
00284 #endif
00285
00286
00287 if (dim_null_ > 0) {
00288     abort_construction = ComputeStencilBoundaryGrid();
00289
00290     #if MTK_DEBUG_LEVEL > 0
00291     if (!abort_construction) {
00292         std::cerr << "Could NOT complete stage 2.4." << std::endl;
00293         std::cerr << "Exiting..." << std::endl;
00294         return false;
00295     }
00296     #endif
00297 }
00298
00299 // Once we have the following three collections of data:
00300 // (a) the coefficients for the interior,
00301 // (b) the coefficients for the boundary (if it applies),
00302 // (c) and the weights (if it applies),
00303 // we will store everything in the output array:
00304
00305 abort_construction = AssembleOperator();
00306
00307 #if MTK_DEBUG_LEVEL > 0
00308 if (!abort_construction) {
00309     std::cerr << "Could NOT complete stage 3." << std::endl;
00310     std::cerr << "Exiting..." << std::endl;
00311     return false;
00312 }
00313 #endif
00314
00315 return true;
00316 }
00317
00318 int mtk::Grad1D::num_bndy_coeffs() const {

```



```

00326
00327     return num_bndy_coeffs_;
00328 }
00329
00330 mtk::Real *mtk::Grad1D::coeffs_interior() const {
00331
00332     return coeffs_interior_;
00333 }
00334
00335 mtk::Real *mtk::Grad1D::weights_crs() const {
00336
00337     return weights_crs_;
00338 }
00339
00340 mtk::Real *mtk::Grad1D::weights_cbs() const {
00341
00342     return weights_cbs_;
00343 }
00344
00345 mtk::DenseMatrix mtk::Grad1D::mim_bndy() const {
00346
00347     mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00348
00349     auto counter = 0;
00350     for (auto ii = 0; ii < dim_null_; ++ii) {
00351         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00352             xx.SetValue(ii, jj, gradient_[2*order_accuracy_ + 1 + counter]);
00353             counter++;
00354         }
00355     }
00356     return xx;
00357 }
00358
00359 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00360     mtk::Real west,
00361                                     mtk::Real east,
00362                                     int num_cells_x) {
00363
00364     int nn{num_cells_x}; // Number of cells on the grid.
00365
00366     #if MTK_DEBUG_LEVEL > 0
00367     mtk::Tools::Prevent(east < west, __FILE__, __LINE__, __func__);
00368     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00369     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00370     #endif
00371
00372     mtk::Real delta_x = (east - west)/((mtk::Real) num_cells_x);
00373
00374     mtk::Real inv_delta_x{mtk::kOne/delta_x};
00375
00376     int gg_num_rows = nn + 1;
00377     int gg_num_cols = nn + 2;
00378     int elements_per_row = num_bndy_coeffs_;
00379     int num_extra_rows = order_accuracy_/2;
00380
00381     // Output matrix featuring sizes for gradient operators.
00382     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00383
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
00400
00401
00402     for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00403         auto jj = ii - num_extra_rows + 1;
00404         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00405             out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00406         }
00407     }

```

```

00408
00410
00411     ee_index = 0;
00412     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00413         auto cc = 0;
00414         for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00415             if(cc >= elements_per_row) {
00416                 out.SetValue(ii,jj,mtk::kZero);
00417             } else {
00418                 out.SetValue(ii,jj,
00419                     -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00420                 cc++;
00421             }
00422         }
00423     }
00424
00425     return out;
00426 }
00427
00428 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(const
    UniStgGrid1D &grid) {
00429
00430     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00431
00432     #if MTK_DEBUG_LEVEL > 0
00433     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00434     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00435     #endif
00436
00437     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00438
00439     int gg_num_rows = nn + 1;
00440     int gg_num_cols = nn + 2;
00441     int elements_per_row = num_bndy_coeffs_;
00442     int num_extra_rows = order_accuracy_/2;
00443
00444     // Output matrix featuring sizes for gradient operators.
00445     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00446
00447
00448
00449     auto ee_index = 0;
00450     for (auto ii = 0; ii < num_extra_rows; ii++) {
00451         auto cc = 0;
00452         for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00453             if(cc >= elements_per_row) {
00454                 out.SetValue(ii, jj, mtk::kZero);
00455             } else {
00456                 out.SetValue(ii,jj,
00457                     gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00458                 cc++;
00459             }
00460         }
00461     }
00462
00463
00464
00465     for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00466         auto jj = ii - num_extra_rows + 1;
00467         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00468             out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00469         }
00470     }
00471
00472
00473
00474     ee_index = 0;
00475     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00476         auto cc = 0;
00477         for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00478             if(cc >= elements_per_row) {
00479                 out.SetValue(ii,jj,mtk::kZero);
00480             } else {
00481                 out.SetValue(ii,jj,
00482                     -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00483                 cc++;
00484             }
00485         }
00486     }
00487
00488     return out;
00489 }
00490
00491 mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix

```

```

(int num_cells_x) {
00492
00493     int nn{num_cells_x}; // Number of cells on the grid.
00494
00495     #if MTK_DEBUG_LEVEL > 0
00496     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00497     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00498     #endif
00499
00500     int gg_num_rows = nn + 1;
00501     int gg_num_cols = nn + 2;
00502     int elements_per_row = num_bndy_coeffs_;
00503     int num_extra_rows = order_accuracy_/2;
00504
00505     // Output matrix featuring sizes for gradient operators.
00506     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00507
00508
00509
00510     auto ee_index = 0;
00511     for (auto ii = 0; ii < num_extra_rows; ii++) {
00512         auto cc = 0;
00513         for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00514             if(cc >= elements_per_row) {
00515                 out.SetValue(ii, jj, mtk::kZero);
00516             } else {
00517                 out.SetValue(ii, jj,
00518                     gradient_[2*order_accuracy_ + 1 + ee_index++]);
00519                 cc++;
00520             }
00521         }
00522     }
00523
00524
00525
00526     for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00527         auto jj = ii - num_extra_rows + 1;
00528         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00529             out.SetValue(ii, jj, coeffs_interior_[cc]);
00530         }
00531     }
00532
00533
00534
00535     ee_index = 0;
00536     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00537         auto cc = 0;
00538         for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00539             if(cc >= elements_per_row) {
00540                 out.SetValue(ii, jj, mtk::kZero);
00541             } else {
00542                 out.SetValue(ii, jj,
00543                     -gradient_[2*order_accuracy_ + 1 + ee_index++]);
00544                 cc++;
00545             }
00546         }
00547     }
00548
00549     return out;
00550 }
00551
00552 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00553
00554
00555
00556     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00557
00558     try {
00559         pp = new mtk::Real[order_accuracy_];
00560     } catch (std::bad_alloc &memory_allocation_exception) {
00561         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00562             std::endl;
00563         std::cerr << memory_allocation_exception.what() << std::endl;
00564     }
00565     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00566
00567     #ifdef MTK_PRECISION_DOUBLE
00568     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00569     #else
00570     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00571     #endif
00572
00573     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00574         pp[ii] = pp[ii - 1] + mtk::kOne;
00575     }

```

```

00576
00577 #if MTK_DEBUG_LEVEL > 0
00578 std::cout << "pp =" << std::endl;
00579 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00580     std::cout << std::setw(12) << pp[ii];
00581 }
00582 std::cout << std::endl << std::endl;
00583 #endif
00584
00585 bool transpose{false};
00586
00587 mtk::DenseMatrix vander_matrix(pp, order_accuracy_, order_accuracy_, transpose);
00588
00589 #if MTK_DEBUG_LEVEL > 0
00590 std::cout << "vander_matrix = " << std::endl;
00591 std::cout << vander_matrix << std::endl << std::endl;
00592 #endif
00593
00594 try {
00595     coeffs_interior_ = new mtk::Real[order_accuracy_];
00596 } catch (std::bad_alloc &memory_allocation_exception) {
00597     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00598         std::endl;
00599     std::cerr << memory_allocation_exception.what() << std::endl;
00600 }
00601 memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00602
00603 coeffs_interior_[1] = mtk::kOne;
00604
00605 #if MTK_DEBUG_LEVEL > 0
00606 std::cout << "oo =" << std::endl;
00607 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00608     std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00609 }
00610 std::cout << std::endl;
00611 #endif
00612
00613 int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00614     coeffs_interior_)};
00615
00616 #if MTK_DEBUG_LEVEL > 0
00617 if (!info) {
00618     std::cout << "System solved! Interior stencil attained!" << std::endl;
00619     std::cout << std::endl;
00620 }
00621 else {
00622     std::cerr << "Something wrong solving system! info = " << info << std::endl;
00623     std::cerr << "Exiting..." << std::endl;
00624     return false;
00625 }
00626 #endif
00627
00628 #if MTK_DEBUG_LEVEL > 0
00629 std::cout << "coeffs_interior_ =" << std::endl;
00630 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00631     std::cout << std::setw(12) << coeffs_interior_[ii];
00632 }
00633 std::cout << std::endl << std::endl;
00634 #endif
00635
00636 delete [] pp;
00637 pp = nullptr;
00638
00639 return true;
00640 }
00641
00642 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00643
00644     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00645
00646     try {
00647         gg = new mtk::Real[num_bndy_coeffs_];
00648     } catch (std::bad_alloc &memory_allocation_exception) {
00649         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00650             std::endl;
00651         std::cerr << memory_allocation_exception.what() << std::endl;
00652     }
00653 }

```

```

00661  memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00662
00663  #ifdef MTK_PRECISION_DOUBLE
00664  gg[1] = 1.0/2.0;
00665  #else
00666  gg[1] = 1.0f/2.0f;
00667  #endif
00668  for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00669      gg[ii] = gg[ii - 1] + mtk::kOne;
00670  }
00671
00672  #if MTK_DEBUG_LEVEL > 0
00673  std::cout << "gg =" << std::endl;
00674  for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00675      std::cout << std::setw(12) << gg[ii];
00676  }
00677  std::cout << std::endl << std::endl;
00678  #endif
00679
00680  bool tran{true}; // Should I transpose the Vandermonde matrix.
00681
00682  mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00683
00684  #if MTK_DEBUG_LEVEL > 0
00685  std::cout << "aa_west_t =" << std::endl;
00686  std::cout << aa_west_t << std::endl;
00687  #endif
00688
00689  mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00690  (aa_west_t));
00691
00692  #if MTK_DEBUG_LEVEL > 0
00693  std::cout << "qq_t =" << std::endl;
00694  std::cout << qq_t << std::endl;
00695  #endif
00696
00697  int kk_num_rows{num_bndy_coeffs_};
00698  int kk_num_cols{dim_null_};
00699
00700  mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00701
00702  // In the case of the gradient, even though we must solve for a null-space
00703  // of dimension 2, we must only extract ONE basis for the kernel.
00704  // We perform this extraction here:
00705
00706  int aux_{kk_num_rows - kk_num_cols};
00707  for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00708      aux_--;
00709      for (auto jj = 0; jj < kk_num_rows; jj++) {
00710          kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =
00711              qq_t.data()[ii*num_bndy_coeffs_ + jj];
00712      }
00713  }
00714
00715  #if MTK_DEBUG_LEVEL > 0
00716  std::cout << "kk =" << std::endl;
00717  std::cout << kk << std::endl;
00718  std::cout << "kk.num_rows() =" << kk.num_rows() << std::endl;
00719  std::cout << "kk.num_cols() =" << kk.num_cols() << std::endl;
00720  std::cout << std::endl;
00721  #endif
00722
00723  // Scale thus requesting that the last entries of the attained basis for the
00724  // null-space, adopt the pattern we require.
00725  // Essentially we will implement the following MATLAB pseudo-code:
00726  // scalers = kk(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00727  // SK = kk*scalers
00728  // where SK is the scaled null-space.
00729
00730  // In this point, we almost have all the data we need correctly allocated
00731  // in memory. We will create the matrix iden_, and elements we wish to scale in
00732  // the kk array. Using the concept of the leading dimension, we could just
00733  // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00734  // GET how does it work. So I will just create a matrix with the content of
00735  // this array that we need, solve for the scalers and then scale the
00736  // whole kk:
00737
00738
00739

```

```

00745 // We will then create memory for that sub-matrix of kk (subk).
00746
00747 mtk::DenseMatrix subk(dim_null_, dim_null_);
00748
00749 auto zz = 0;
00750 for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00751     for (auto jj = 0; jj < dim_null_; jj++) {
00752         subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00753     }
00754     zz++;
00755 }
00756
00757 #if MTK_DEBUG_LEVEL > 0
00758 std::cout << "subk =" << std::endl;
00759 std::cout << subk << std::endl;
00760 #endif
00761
00762 subk.Transpose();
00763
00764 #if MTK_DEBUG_LEVEL > 0
00765 std::cout << "subk_t =" << std::endl;
00766 std::cout << subk << std::endl;
00767 #endif
00768
00769 bool padded{false};
00770 tran = false;
00771
00772 mtk::DenseMatrix iden(dim_null_, padded, tran);
00773
00774 #if MTK_DEBUG_LEVEL > 0
00775 std::cout << "iden =" << std::endl;
00776 std::cout << iden << std::endl;
00777 #endif
00778
00779 // Solve the system to compute the scalars.
00780 // An example of the system to solve, for k = 8, is:
00781 //
00782 // subk*scalars = iden or
00783 //
00784 // | 0.386018 -0.0339244 -0.129478 | | 1 0 0 |
00785 // | -0.119774 0.0199423 0.0558632 |*scalars = | 0 1 0 |
00786 // | 0.0155708 -0.00349546 -0.00853182 | | 0 0 1 |
00787 //
00788 // Notice this is a nrhs = 3 system.
00789 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalars... they
00790 // will be stored in the created identity matrix.
00791 // Let us first transpose subk (because of LAPACK):
00792
00793 int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00794
00795 #if MTK_DEBUG_LEVEL > 0
00796 if (!info) {
00797     std::cout << "System successfully solved!" <<
00798         std::endl;
00799 } else {
00800     std::cerr << "Something went wrong solving system! info = " << info <<
00801         std::endl;
00802     std::cerr << "Exiting..." << std::endl;
00803     return false;
00804 }
00805 std::cout << std::endl;
00806 #endif
00807
00808 #if MTK_DEBUG_LEVEL > 0
00809 std::cout << "Computed scalars:" << std::endl;
00810 std::cout << iden << std::endl;
00811 #endif
00812
00813 // Multiply the two matrices to attain a scaled basis for null-space.
00814
00815 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00816
00817 #if MTK_DEBUG_LEVEL > 0
00818 std::cout << "Rational basis for the null-space:" << std::endl;
00819 std::cout << rat_basis_null_space_ << std::endl;
00820 #endif
00821
00822 // At this point, we have a rational basis for the null-space, with the
00823 // pattern we need! :)
00824
00825 delete [] gg;

```

```

00826     gg = nullptr;
00827
00828     return true;
00829 }
00830
00831 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00832
00833
00834
00835     mtk::Real *gg{}; // Generator vector for the first approximation.
00836
00837     try {
00838         gg = new mtk::Real[num_bndy_coeffs_];
00839     } catch (std::bad_alloc &memory_allocation_exception) {
00840         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00841             std::endl;
00842         std::cerr << memory_allocation_exception.what() << std::endl;
00843     }
00844     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00845
00846     #ifdef MTK_PRECISION_DOUBLE
00847         gg[1] = 1.0/2.0;
00848     #else
00849         gg[1] = 1.0f/2.0f;
00850     #endif
00851     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00852         gg[ii] = gg[ii - 1] + mtk::kOne;
00853     }
00854
00855     #if MTK_DEBUG_LEVEL > 0
00856     std::cout << "gg0 =" << std::endl;
00857     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00858         std::cout << std::setw(12) << gg[ii];
00859     }
00860     std::cout << std::endl << std::endl;
00861     #endif
00862
00863     // Allocate 2D array to store the collection of preliminary approximations.
00864     try {
00865         prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00866     } catch (std::bad_alloc &memory_allocation_exception) {
00867         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00868             std::endl;
00869         std::cerr << memory_allocation_exception.what() << std::endl;
00870     }
00871     memset(prem_apps_,
00872         mtk::kZero,
00873         sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00874
00875
00876
00877     for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00878
00879         // Re-check new generator vector for every iteration except for the first.
00880         #if MTK_DEBUG_LEVEL > 0
00881         if (ll > 0) {
00882             std::cout << "gg" << ll << " =" << std::endl;
00883             for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00884                 std::cout << std::setw(12) << gg[ii];
00885             }
00886             std::cout << std::endl << std::endl;
00887         }
00888         #endif
00889
00890
00891
00892         bool transpose{false};
00893
00894         mtk::DenseMatrix aa(gg,
00895             num_bndy_coeffs_, order_accuracy_ + 1,
00896             transpose);
00897
00898         #if MTK_DEBUG_LEVEL > 0
00899         std::cout << "aa_" << ll << " =" << std::endl;
00900         std::cout << aa << std::endl;
00901         #endif
00902
00903
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_DEBUG_LEVEL > 0
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     std::cout << std::endl;
00926     #endif
00927
00928
00929     // However, this is an under-determined system of equations. So we can not
00930     // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00931     // our LAPACKAdapter class.
00932
00933     int info_{
00934         mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
00935             , ob_ld)};
00936
00937     #if MTK_DEBUG_LEVEL > 0
00938     if (!info_) {
00939         std::cout << "System successfully solved!" << std::endl << std::endl;
00940     } else {
00941         std::cerr << "Error solving system! info = " << info_ << std::endl;
00942     }
00943     #endif
00944
00945     #if MTK_DEBUG_LEVEL > 0
00946     std::cout << "ob =" << std::endl;
00947     for (auto ii = 0; ii < ob_ld; ++ii) {
00948         std::cout << std::setw(12) << ob[ii] << std::endl;
00949     }
00950     std::cout << std::endl;
00951     #endif
00952
00953
00954     // This implies a DAXPY operation. However, we must construct the arguments
00955     // for this operation.
00956
00957     // Save them into the ob_bottom array:
00958
00959     Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00960
00961     try {
00962         ob_bottom = new mtk::Real[dim_null_];
00963     } catch (std::bad_alloc &memory_allocation_exception) {
00964         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00965             std::endl;
00966         std::cerr << memory_allocation_exception.what() << std::endl;
00967     }
00968     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00969
00970     for (auto ii = 0; ii < dim_null_; ++ii) {
00971         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00972     }
00973
00974     #if MTK_DEBUG_LEVEL > 0
00975     std::cout << "ob_bottom =" << std::endl;
00976     for (auto ii = 0; ii < dim_null_; ++ii) {
00977         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00978     }
00979     std::cout << std::endl;
00980     #endif
00981
00982
00983     // We must computed an scaled ob, sob, using the scaled null-space in
00984     // rat_basis_null_space_.
00985     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00986     // or:
00987     // thus:
00988     //      Y =      a*A      *x      +      b*Y (DAXPY).
00989
00990     #if MTK_DEBUG_LEVEL > 0
00991     std::cout << "Rational basis for the null-space:" << std::endl;
00992     std::cout << rat_basis_null_space_ << std::endl;
00993
00994

```



```

00995     #endif
00996
00997     mtk::Real alpha{~mtk::kOne};
00998     mtk::Real beta{mtk::kOne};
00999
01000     mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
01001                                   ob_bottom, beta, ob);
01002
01003     #if MTK_DEBUG_LEVEL > 0
01004     std::cout << "scaled ob:" << std::endl;
01005     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01006         std::cout << std::setw(12) << ob[ii] << std::endl;
01007     }
01008     std::cout << std::endl;
01009     #endif
01010
01011     // We save the recently scaled solution, into an array containing these.
01012     // We can NOT start building the pi matrix, simply because I want that part
01013     // to be separated since its construction depends on the algorithm we want
01014     // to implement.
01015
01016     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01017         prem_apps_[ii*num_bndy_approxs_ + 11] = ob[ii];
01018     }
01019
01020     // After the first iteration, simply shift the entries of the last
01021     // generator vector used:
01022     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01023         gg[ii]--;
01024     }
01025
01026     // Garbage collection for this loop:
01027     delete[] ob;
01028     ob = nullptr;
01029
01030     delete[] ob_bottom;
01031     ob_bottom = nullptr;
01032 } // End of: for (ll = 0; ll < dim_null; ll++);
01033
01034 #if MTK_DEBUG_LEVEL > 0
01035 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
01036 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01037     for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01038         std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
01039     }
01040     std::cout << std::endl;
01041 }
01042 std::cout << std::endl;
01043 #endif
01044
01045 delete[] gg;
01046 gg = nullptr;
01047
01048 return true;
01049 }
01050
01051 bool mtk::Grad1D::ComputeWeights() {
01052
01053     // Matrix to compute the weights as in the CRSA.
01054     mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
01055
01056     // Assemble the pi matrix using:
01057     // 1. The collection of scaled preliminary approximations.
01058     // 2. The collection of coefficients approximating at the interior.
01059     // 3. The scaled basis for the null-space.
01060
01061     // 1.1. Process array of scaled preliminary approximations.
01062
01063     // These are queued in scaled_solutions. Each one of these, will be a column
01064     // of the pi matrix:
01065     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01066         for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01067             pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01068                 prem_apps_[ii*num_bndy_approxs_ + jj];
01069         }
01070     }
01071
01072     // 1.2. Add columns from known stencil approximating at the interior.
01073
01074     // However, these must be padded by zeros, according to their position in the

```

```

01077 // final pi matrix:
01078 auto mm = 1;
01079 for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
01080     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01081         auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
01082             (order_accuracy_/2 + 1)) + jj;
01083         pi.data()[de] = coeffs_interior_[ii];
01084     }
01085     ++mm;
01086 }
01087
01088 rat_basis_null_space_.OrderColMajor();
01089
01090 #if MTK_DEBUG_LEVEL > 0
01091 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01092 std::cout << rat_basis_null_space_ << std::endl;
01093 #endif
01094
01095 // 1.3. Add final set of columns: rational basis for null-space.
01096
01097 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01098     jj < num_bndy_coeffs_ - 1; ++jj) {
01099     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01100         auto og =
01101             (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01102         auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01103         pi.data()[de] = rat_basis_null_space_.data()[og];
01104     }
01105 }
01106
01107 #if MTK_DEBUG_LEVEL > 0
01108 std::cout << "coeffs_interior_ =" << std::endl;
01109 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01110     std::cout << std::setw(12) << coeffs_interior_[ii];
01111 }
01112 std::cout << std::endl << std::endl;
01113 #endif
01114
01115 #if MTK_DEBUG_LEVEL > 0
01116 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01117 std::cout << pi << std::endl;
01118 #endif
01119
01120 // This imposes the mimetic condition.
01121
01122 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01123
01124 try {
01125     hh = 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(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01132
01133 hh[0] = -mtk::kOne;
01134 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01135     auto aux_xx = mtk::kZero;
01136     for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01137         aux_xx += coeffs_interior_[jj];
01138     }
01139     hh[ii] = -mtk::kOne*aux_xx;
01140 }
01141
01142 // That is, we construct a system, to solve for the weights.
01143
01144 // Once again we face the challenge of solving with LAPACK. However, for the
01145 // CRS, this matrix PI is over-determined, since it has more rows than
01146 // unknowns. However, according to the theory, the solution to this system is
01147 // unique. We will use dgels_.
01148
01149 try {
01150     weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01151 } catch (std::bad_alloc &memory_allocation_exception) {
01152     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01153         std::endl;
01154     std::cerr << memory_allocation_exception.what() << std::endl;
01155 }

```

```

01160  memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01161
01162  int weights_ld{pi.num_cols() + 1};
01163
01164  // Preserve hh.
01165  std::copy(hh, hh + weights_ld, weights_cbs_);
01166
01167  pi.Transpose();
01168
01169  int info{
01170      mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01171          weights_cbs_, weights_ld)
01172  };
01173
01174  #if MTK_DEBUG_LEVEL > 0
01175  if (!info) {
01176      std::cout << "System successfully solved!" << std::endl << std::endl;
01177  } else {
01178      std::cerr << "Error solving system! info = " << info << std::endl;
01179  }
01180  #endif
01181
01182  #if MTK_DEBUG_LEVEL > 0
01183  std::cout << "hh =" << std::endl;
01184  for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01185      std::cout << std::setw(11) << hh[ii] << std::endl;
01186  }
01187  std::cout << std::endl;
01188  #endif
01189
01190  // Preserve the original weights for research.
01191
01192  try {
01193      weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01194  } catch (std::bad_alloc &memory_allocation_exception) {
01195      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01196          std::endl;
01197      std::cerr << memory_allocation_exception.what() << std::endl;
01198  }
01199  memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01200
01201  std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01202
01203  #if MTK_DEBUG_LEVEL > 0
01204  std::cout << "weights_CRSA + lambda =" << std::endl;
01205  for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01206      std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01207  }
01208  std::cout << std::endl;
01209  #endif
01210
01211
01212
01213  if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01214
01215      int minrow_{std::numeric_limits<int>::infinity()};
01216
01217      mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01218          order_accuracy_)};
01219      mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01220
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_ + 1) + 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++) {
01259         if (ii == 0) {
01260             phi.data()[jj] = 0.0;
01261         } else {
01262             phi.data()[ (ii + mm)*order_accuracy_ + jj] = coeffs_interior_[ii - 1];
01263         }
01264     }
01265     mm++;
01266 }
01267
01268 #if MTK_DEBUG_LEVEL > 0
01269 std::cout << "phi =" << std::endl;
01270 std::cout << phi << std::endl;
01271 #endif
01272
01273 mtk::Real *lamed{}; // Used to build big lambda.
01274
01275 try {
01276     lamed = new mtk::Real[num_bndy_approxs_ - 1];
01277 } catch (std::bad_alloc &memory_allocation_exception) {
01278     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01279         std::endl;
01280     std::cerr << memory_allocation_exception.what() << std::endl;
01281 }
01282 memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approxs_ - 1));
01283
01284 for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01285     lamed[ii] = hh[ii + order_accuracy_ + 1];
01286 }
01287
01288 #if MTK_DEBUG_LEVEL > 0
01289 std::cout << "lamed =" << std::endl;
01290 for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01291     std::cout << std::setw(12) << lamed[ii] << std::endl;
01292 }
01293 std::cout << std::endl;
01294 #endif
01295
01296 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01297     mtk::Real temp = mtk::kZero;
01298     for (auto jj = 0; jj < num_bndy_approxs_ - 1; ++jj) {
01299         temp = temp +
01300             lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01301     }
01302     hh[ii] = hh[ii] - temp;
01303 }
01304
01305 #if MTK_DEBUG_LEVEL > 0
01306 std::cout << "big_lambda =" << std::endl;
01307 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01308     std::cout << std::setw(12) << hh[ii] << std::endl;
01309 }
01310 std::cout << std::endl;
01311 #endif
01312
01313 int copy_result{}; // Should I replace the solution... not for now.
01314
01315 mtk::Real normerr_; // Norm of the error for the solution on each row.
01316
01317 for (auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01318     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01319         data(),
01320         order_accuracy_ + 1,

```

```

01324                                     order_accuracy_,
01325                                     order_accuracy_,
01326                                     hh,
01327                                     weights_cbs_,
01328                                     row_,
01329                                     mimetic_threshold_,
01330                                     copy_result);
01331     mtk::Real aux{normerr_/norm};
01332
01333     #if MTK_DEBUG_LEVEL>0
01334     std::cout << "Relative norm: " << aux << " " << std::endl;
01335     std::cout << std::endl;
01336     #endif
01337
01338     if (aux < minnorm) {
01339         minnorm = aux;
01340         minrow_ = row_;
01341     }
01342 }
01343
01344 #if MTK_DEBUG_LEVEL > 0
01345 std::cout << "weights_CBSA + lambda (after brute force search):" <<
01346 std::endl;
01347 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01348     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01349 }
01350 std::cout << std::endl;
01351 #endif
01352
01353 // After we know which row yields the smallest relative norm that row is
01354 // chosen to be the objective function and the result of the optimizer is
01355 // chosen to be the new weights_.
01356
01357 #if MTK_DEBUG_LEVEL > 0
01358 std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01359 minrow_ + 1 << std::endl;
01360 std::cout << std::endl;
01361 #endif
01362
01363 copy_result = 1;
01364 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01365                                     order_accuracy_ + 1,
01366                                     order_accuracy_,
01367                                     order_accuracy_,
01368                                     hh,
01369                                     weights_cbs_,
01370                                     minrow_,
01371                                     mimetic_threshold_,
01372                                     copy_result);
01373
01374 mtk::Real aux_{normerr_/norm};
01375 #if MTK_DEBUG_LEVEL > 0
01376 std::cout << "Relative norm: " << aux_ << std::endl;
01377 std::cout << std::endl;
01378 #endif
01379
01380 delete [] lamed;
01381 lamed = nullptr;
01382 }
01383
01384 delete [] hh;
01385 hh = nullptr;
01386
01387 return true;
01388 }
01389
01390 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01391 #if MTK_DEBUG_LEVEL > 0
01392 std::cout << "weights_* + lambda =" << std::endl;
01393 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01394     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01395 }
01396 std::cout << std::endl;
01397 #endif
01398
01399 mtk::Real *lambda{}; // Collection of bottom values from weights_.
01400
01401 try {

```

```

01406     lambda = new mtk::Real[dim_null_];
01407 } catch (std::bad_alloc &memory_allocation_exception) {
01408     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01409         std::endl;
01410     std::cerr << memory_allocation_exception.what() << std::endl;
01411 }
01412 memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01413
01414 for (auto ii = 0; ii < dim_null_; ++ii) {
01415     lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01416 }
01417
01418 #if MTK_DEBUG_LEVEL > 0
01419 std::cout << "lambda =" << std::endl;
01420 for (auto ii = 0; ii < dim_null_; ++ii) {
01421     std::cout << std::setw(12) << lambda[ii] << std::endl;
01422 }
01423 std::cout << std::endl;
01424 #endif
01425
01426 mtk::Real *alpha{}; // Collection of alpha values.
01427
01428 try {
01429     alpha = new mtk::Real[dim_null_];
01430 } catch (std::bad_alloc &memory_allocation_exception) {
01431     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01432         std::endl;
01433     std::cerr << memory_allocation_exception.what() << std::endl;
01434 }
01435 memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01436
01437 for (auto ii = 0; ii < dim_null_; ++ii) {
01438     alpha[ii] = lambda[ii]/weights_cbs_[ii];
01439 }
01440
01441 #if MTK_DEBUG_LEVEL > 0
01442 std::cout << "alpha =" << std::endl;
01443 for (auto ii = 0; ii < dim_null_; ++ii) {
01444     std::cout << std::setw(12) << alpha[ii] << std::endl;
01445 }
01446 std::cout << std::endl;
01447 #endif
01448
01449 try {
01450     mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01451 } catch (std::bad_alloc &memory_allocation_exception) {
01452     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01453         std::endl;
01454     std::cerr << memory_allocation_exception.what() << std::endl;
01455 }
01456 memset(mim_bndy_,
01457     mtk::kZero,
01458     sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01459
01460 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01461     for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {
01462         mim_bndy_[ii*num_bndy_approxs_ + jj] =
01463             prem_apps_[ii*num_bndy_approxs_ + jj] +
01464             alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01465     }
01466 }
01467
01468 for(auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01469     mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01470         prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01471 }
01472
01473 #if MTK_DEBUG_LEVEL > 0
01474 std::cout << "Collection of mimetic approximations:" << std::endl;
01475 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01476     for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01477         std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01478     }
01479     std::cout << std::endl;
01480 }
01481 std::cout << std::endl;
01482 #endif
01483
01484 delete[] lambda;

```

```

01489     lambda = nullptr;
01490
01491     delete[] alpha;
01492     alpha = nullptr;
01493
01494     return true;
01495 }
01496
01497 bool mtk::Grad1D::AssembleOperator(void) {
01498
01499     // The output array will have this form:
01500     // 1. The first entry of the array will contain the used order kk.
01501     // 2. The second entry of the array will contain the collection of
01502     // approximating coefficients for the interior of the grid.
01503     // 3. The third entry will contain a collection of weights.
01504     // 4. The next dim_null - 1 entries will contain the collections of
01505     // approximating coefficients for the west boundary of the grid.
01506
01507     gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01508         num_bndy_approxs_*num_bndy_coeffs_;
01509
01510     #if MTK_DEBUG_LEVEL > 0
01511     std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01512     #endif
01513
01514     try {
01515         gradient_ = new mtk::Real[gradient_length_];
01516     } catch (std::bad_alloc &memory_allocation_exception) {
01517         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01518             std::endl;
01519         std::cerr << memory_allocation_exception.what() << std::endl;
01520     }
01521     memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01522
01523     gradient_[0] = order_accuracy_;
01524
01525     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01526         gradient_[ii + 1] = coeffs_interior_[ii];
01527     }
01528
01529     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01530         gradient_[order_accuracy_ + 1 + ii] = weights_cbs_[ii];
01531     }
01532
01533     int offset{2*order_accuracy_ + 1};
01534
01535     int aux {}; // Auxiliary variable.
01536
01537     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01538         for (auto ii = 0; ii < num_bndy_approxs_ ; ii++) {
01539             for (auto jj = 0; jj < num_bndy_coeffs_ ; jj++) {
01540                 gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01541                 aux++;
01542             }
01543         }
01544     } else {
01545         gradient_[offset + 0] = prem_apps_[0];
01546         gradient_[offset + 1] = prem_apps_[1];
01547         gradient_[offset + 2] = prem_apps_[2];
01548     }
01549
01550     #if MTK_DEBUG_LEVEL > 0
01551     std::cout << "1D " << order_accuracy_ << "--order grad built!" << std::endl;
01552     std::cout << std::endl;
01553     #endif
01554
01555     return true;
01556 }

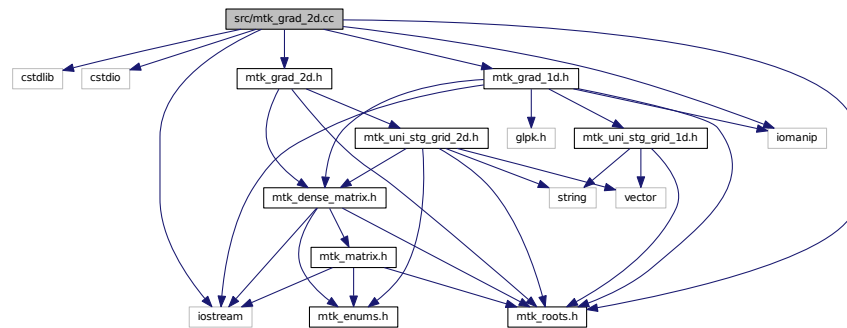
```

17.69 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:



17.69.1 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](#).

17.70 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,
                                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
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     if (!info) {
00094         std::cerr << "Mimetic grad could not be built." << std::endl;
00095         return info;
00096     }
00097
00098     auto west = grid.west_bndy();
00099     auto east = grid.east_bndy();
00100     auto south = grid.south_bndy();
00101     auto north = grid.east_bndy();
00102
00103     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00104     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00105
00106     mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00107     mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00108
00109     bool padded{true};
00110     bool transpose{true};
00111
00112     mtk::DenseMatrix tix(num_cells_x, padded, transpose);

```

```

00113 mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00114
00115 mtk::DenseMatrix gxy(mtk::DenseMatrix::Kron(tiy, Gx));
00116 mtk::DenseMatrix gyx(mtk::DenseMatrix::Kron(Gy, tix));
00117
00118 #if MTK_DEBUG_LEVEL > 0
00119 std::cout << "Gx :" << mx << "by " << nx << std::endl;
00120 std::cout << "Transpose Iy : " << num_cells_y << " by " << ny << std::endl;
00121 std::cout << "Gy :" << my << "by " << ny << std::endl;
00122 std::cout << "Transpose Ix : " << num_cells_x << " by " << nx << std::endl;
00123 std::cout << "Kronecker dimensions Grad 2D" <<
00124     mx*num_cells_y + my*num_cells_x << " by " << nx*ny << std::endl;
00125 #endif
00126
00127 mtk::DenseMatrix g2d(mx*num_cells_y + my*num_cells_x, nx*ny);
00128
00129 for(auto ii = 0; ii < nx*ny; ii++) {
00130     for(auto jj = 0; jj < mx*num_cells_y; jj++) {
00131         g2d.SetValue(jj,ii, gxy.GetValue(jj,ii));
00132     }
00133     for(auto kk = 0; kk < my*num_cells_x; kk++) {
00134         g2d.SetValue(kk + mx*num_cells_y, ii, gyx.GetValue(kk,ii));
00135     }
00136 }
00137
00138 gradient_ = g2d;
00139
00140 return info;
00141 }
00142
00143 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix() {
00144
00145     return gradient_;
00146 }

```

17.71 src/mtk_interp_1d.cc File Reference

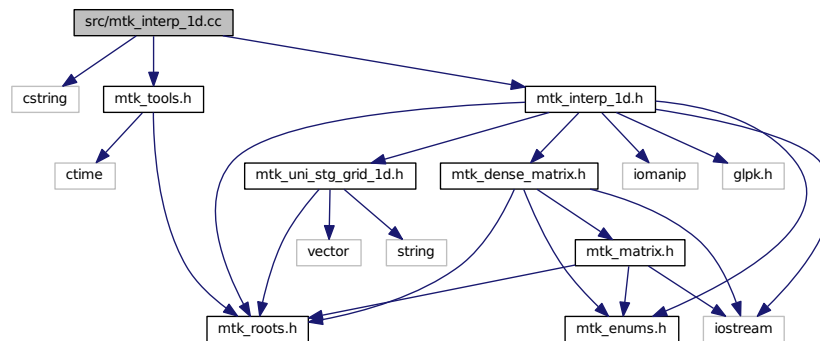
Includes the implementation of the class Interp1D.

```

#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)`

17.71.1 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](#).

17.72 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.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 #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
00069     stream << "coeffs_interior_[1:" << in.order_accuracy_ << "]" = ";
00070     for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00071         stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00072     }
00073     stream << std::endl;
00074
00075     return stream;
00076 }
00077
00078 }
00079
00080 mtk::Interp1D::Interp1D():
00081     dir_interp_(mtk::SCALAR_TO_VECTOR),
00082     order_accuracy_(mtk::kDefaultOrderAccuracy),
00083     coeffs_interior_(nullptr) {}
00084
00085 mtk::Interp1D::Interp1D(const Interp1D &interp):
00086     dir_interp_(interp.dir_interp_),
00087     order_accuracy_(interp.order_accuracy_),
00088     coeffs_interior_(interp.coeffs_interior_) {}
00089
00090 mtk::Interp1D::~Interp1D() {
00091
00092     delete[] coeffs_interior_;
00093     coeffs_interior_ = nullptr;
00094 }
00095
00096 bool mtk::Interp1D::ConstructInterp1D(int order_accuracy,
00097     mtk::DirInterp dir) {
00098
00099     #if MTK_DEBUG_LEVEL > 0
00100     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00101     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00102     mtk::Tools::Prevent(dir < mtk::SCALAR_TO_VECTOR &&
00103         dir > mtk::VECTOR_TO_SCALAR,
00104         __FILE__, __LINE__, __func__);
00105
00106     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00107     #endif
00108
00109     order_accuracy_ = order_accuracy;
00110
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
00127     return true;
00128 }
00129
00130 mtk::Real *mtk::Interp1D::coeffs_interior() const {
00131
00132     return coeffs_interior_;
00133 }
00134
00135 mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix(const
00136     UniStgGrid1D &grid) {
00137
00138     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00139
00140     #if MTK_DEBUG_LEVEL > 0

```

```

00140     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00141     #endif
00142
00143     int gg_num_rows{}; // Number of rows.
00144     int gg_num_cols{}; // Number of columns.
00145
00146     if (dir_interp_ == mtk::SCALAR_TO_VECTOR) {
00147         gg_num_rows = nn + 1;
00148         gg_num_cols = nn + 2;
00149     } else {
00150         gg_num_rows = nn + 2;
00151         gg_num_cols = nn + 1;
00152     }
00153
00154     // Output matrix featuring sizes for gradient operators.
00155
00156     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00157
00158
00159     out.SetValue(0, 0, mtk::kOne);
00160
00161
00162     for (auto ii = 1; ii < gg_num_rows - 1; ++ii) {
00163         for (auto jj = ii; jj < order_accuracy_ + ii; ++jj) {
00164             out.SetValue(ii, jj, mtk::kOne/order_accuracy_);
00165         }
00166     }
00167
00168
00169     out.SetValue(gg_num_rows - 1, gg_num_cols - 1, mtk::kOne);
00170
00171     return out;
00172 }

```

17.73 src/mtk_lap_1d.cc File Reference

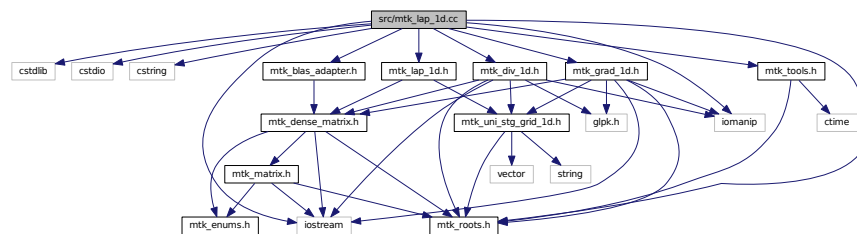
Includes the implementation of the class Lap1D.

```

#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)`

17.73.1 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](#).

17.74 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_1d.h"
00068 #include "mtk_div_1d.h"
00069 #include "mtk_lap_1d.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;
00089     for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00090         for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00091             stream << std::setw(13) <<
00092                 in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj];
00093         }
00094         stream << std::endl;
00095     }
00096     return stream;
00097 }
00098
00099 mtk::Lap1D::Lap1D():
00100     order_accuracy_(mtk::kDefaultOrderAccuracy),
00101     laplacian_length_(),
00102     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00103
00104 mtk::Lap1D::~~Lap1D() {
00105     delete [] laplacian_;
00106     laplacian_ = nullptr;
00107 }
00108
00109 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00110     mtk::Real mimetic_threshold) {
00111
00112     #if MTK_DEBUG_LEVEL > 0
00113     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00114     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00115     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00116         __FILE__, __LINE__, __func__);
00117     #endif
00118     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00119         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00120     }
00121
00122     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00123     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00124     #endif
00125     order_accuracy_ = order_accuracy;

```

```

00137 mimetic_threshold_ = mimetic_threshold;
00138
00140
00141 mtk::Grad1D grad; // Mimetic gradient.
00142
00143 bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00144
00145 if (!info) {
00146     std::cerr << "Mimetic grad could not be built." << std::endl;
00147     return false;
00148 }
00149
00151 mtk::Div1D div; // Mimetic divergence.
00152
00153 info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00154
00155 if (!info) {
00156     std::cerr << "Mimetic div could not be built." << std::endl;
00157     return false;
00158 }
00159
00160
00162 // Since these are mimetic operator, we must multiply the matrices arising
00164 // from both the divergence and the Laplacian, in order to get the
00165 // approximating coefficients for the Laplacian operator.
00166
00167 // However, we must choose a grid that implied a step size of 1, so to get
00168 // the approximating coefficients, without being affected from the
00169 // normalization with respect to the grid.
00170
00171 // Also, the grid must be of the minimum size to support the requested order
00172 // of accuracy. We must please the divergence.
00173
00174 mtk::UniStgGrid1D aux(mtk::kZero,
00175                       (mtk::Real) 3*order_accuracy_ - 1,
00176                       3*order_accuracy_ - 1);
00177
00178 #if MTK_DEBUG_LEVEL > 0
00179 std::cout << "aux =" << std::endl;
00180 std::cout << aux << std::endl;
00181 std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00182 std::cout << std::endl;
00183 #endif
00184
00185 mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00186
00187 #if MTK_DEBUG_LEVEL > 0
00188 std::cout << "grad_m =" << std::endl;
00189 std::cout << grad_m << std::endl;
00190 #endif
00191
00192 mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux));
00193
00194 #if MTK_DEBUG_LEVEL > 0
00195 std::cout << "div_m =" << std::endl;
00196 std::cout << div_m << std::endl;
00197 #endif
00198
00202 mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00204
00205 lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00206
00207 #if MTK_DEBUG_LEVEL > 0
00208 std::cout << "lap =" << std::endl;
00209 std::cout << lap << std::endl;
00210 #endif
00211
00213 // The output array will have this form:
00216 // 1. The first entry of the array will contain the used order kk.
00218 // 2. The second entry of the array will contain the collection of
00219 // approximating coefficients for the interior of the grid.
00220 // 3. The next entries will contain the collections of approximating
00221 // coefficients for the west boundary of the grid.
00222
00223 laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00224     (order_accuracy_ - 1)*(2*order_accuracy_);
00225

```



```

00226  #if MTK_DEBUG_LEVEL > 0
00227  std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00228  std::cout << std::endl;
00229  #endif
00230
00231  try {
00232      laplacian_ = new mtk::Real[laplacian_length_];
00233  } catch (std::bad_alloc &memory_allocation_exception) {
00234      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00235          std::endl;
00236      std::cerr << memory_allocation_exception.what() << std::endl;
00237  }
00238  memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00239
00241  laplacian_[0] = order_accuracy_;
00243
00246  for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00247      laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00249  }
00250
00252  auto offset = 1 + (2*order_accuracy_ - 1);
00254  for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00255      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00256          laplacian_[offset + ii*(2*order_accuracy_) + jj] =
00257              lap.GetValue(1 + ii, jj);
00259      }
00260  }
00261
00262  return true;
00263 }
00264
00265 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(const
    UniStgGrid1D &grid) {
00266
00267     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00268
00269     #if MTK_DEBUG_LEVEL > 0
00270     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00271     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00272     #endif
00273
00274     mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00275
00276     mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
    dx^2.
00277
00279     auto offset = (1 + 2*order_accuracy_ - 1);
00281     for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00282         for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00283             lap.SetValue(1 + ii,
00284                 jj,
00285                 idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00287         }
00288     }
00289
00291     offset = 1 + (order_accuracy_ - 1);
00292
00293     int kk{1};
00294     for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00295         int mm{1};
00296         for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00297             lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00298             mm = mm + 1;
00299         }
00300         kk = kk + 1;
00301     }
00302
00303
00305     offset = (1 + 2*order_accuracy_ - 1);
00307     auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00309
00310     auto ll = 1;
00311     auto rr = 1;

```

```

00312     for (auto ii = nn; ii > aux - 1; --ii) {
00313         auto cc = 0;
00314         for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00315             lap.SetValue(ii, jj, lap.GetValue(rr, cc));
00316             ++ll;
00317             ++cc;
00318         }
00319         rr++;
00320     }
00321
00322     return lap;
00323 }
00324
00325 mtk::Real* mtk::Lap1D::data(const UniStgGrid1D &grid) {
00326     mtk::DenseMatrix tmp;
00327     tmp = ReturnAsDenseMatrix(grid);
00328     return tmp.data();
00329 }

```

17.75 src/mtk_lap_2d.cc File Reference

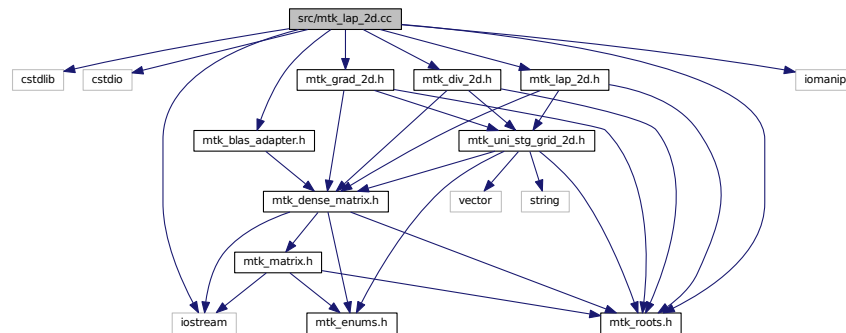
Includes the implementation of the class Lap2D.

```

#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:



17.75.1 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](#).

17.76 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     int num_cells_x{grid.num_cells_x()};
00082     int num_cells_y{grid.num_cells_y()};
00083     int aux{(num_cells_x + 2)*(num_cells_y + 2)};
00084
00085     mtk::Grad2D gg;
00086     mtk::Div2D dd;
00087
00088     bool info{gg.ConstructGrad2D(grid, order_accuracy, mimetic_threshold)};
00089
00090     if (!info) {
00091         std::cerr << "Mimetic lap could not be built." << std::endl;
00092         return info;
00093     }
00094
00095     info = dd.ConstructDiv2D(grid, order_accuracy, mimetic_threshold);
00096
00097     if (!info) {
00098         std::cerr << "Mimetic div could not be built." << std::endl;
00099         return info;
00100     }
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() {
00111
00112     return laplacian_;
00113 }
00114
00115 mtk::Real* mtk::Lap2D::data() {
00116
00117     return laplacian_.data();
00118 }

```

17.77 src/mtk_lapack_adapter.cc File Reference

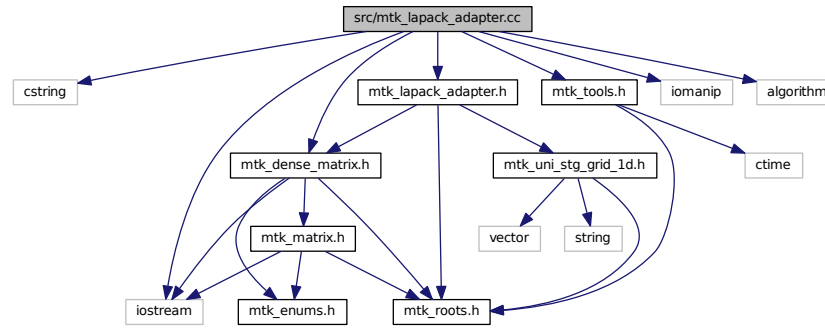
Adapter class for the LAPACK API.

```

#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.

17.77.1 Detailed Description

This class contains a collection of static classes, 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](#).

17.78 mtk_lapack_adapter.cc

```

00001
00021 /*
00022 Copyright (C) 2015, Computational Science Research Center, San Diego State
00023 University. All rights reserved.
00024
00025 Redistribution and use in source and binary forms, with or without modification,
00026 are permitted provided that the following conditions are met:
00027
00028 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00029 and a copy of the modified files should be reported once modifications are
00030 completed, unless these modifications are made through the project's GitHub
00031 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00032 should be developed and included in any deliverable.
00033
00034 2. Redistributions of source code must be done through direct
00035 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00036
00037 3. Redistributions in binary form must reproduce the above copyright notice,
00038 this list of conditions and the following disclaimer in the documentation and/or
00039 other materials provided with the distribution.
00040
00041 4. Usage of the binary form on proprietary applications shall require explicit
00042 prior written permission from the the copyright holders, and due credit should
00043 be given to the copyright holders.
00044
00045 5. Neither the name of the copyright holder nor the names of its contributors
00046 may be used to endorse or promote products derived from this software without
00047 specific prior written permission.
00048
00049 The copyright holders provide no reassurances that the source code provided does
00050 not infringe any patent, copyright, or any other intellectual property rights of
00051 third parties. The copyright holders disclaim any liability to any recipient for
00052 claims brought against recipient by any third party for infringement of that
00053 parties intellectual property rights.
00054
00055 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00056 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00057 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00058 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00059 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00060 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00061 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00062 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00063 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00064 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00065 */
00066
00067 #include <cstring>
00068
00069 #include <iostream>
00070 #include <iomanip>
00071
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075 #include "mtk_dense_matrix.h"
00076 #include "mtk_lapack_adapter.h"
00077
00078 namespace mtk {
00079
00080 extern "C" {
00081
00082 #ifdef MTK_PRECISION_DOUBLE
00083
00102 void dgesv_(int* n,
00103             int* nrhs,
00104             Real* a,
00105             int* lda,

```

```
00106         int* ipiv,
00107         Real* b,
00108         int* ldb,
00109         int* info);
00110 #else
00111
00130 void sgesv_(int* n,
00131             int* nrhs,
00132             Real* a,
00133             int* lda,
00134             int* ipiv,
00135             Real* b,
00136             int* ldb,
00137             int* info);
00138 #endif
00139
00140 #ifdef MTK_PRECISION_DOUBLE
00141
00184 void dgels_(char* trans,
00185             int* m,
00186             int* n,
00187             int* nrhs,
00188             Real* a,
00189             int* lda,
00190             Real* b,
00191             int* ldb,
00192             Real* work,
00193             int* lwork,
00194             int* info);
00195 #else
00196
00239 void sgels_(char* trans,
00240             int* m,
00241             int* n,
00242             int* nrhs,
00243             Real* a,
00244             int* lda,
00245             Real* b,
00246             int* ldb,
00247             Real* work,
00248             int* lwork,
00249             int* info);
00250 #endif
00251
00252 #ifdef MTK_PRECISION_DOUBLE
00253
00282 void dgeqrf_(int *m,
00283              int *n,
00284              Real *a,
00285              int *lda,
00286              Real *tau,
00287              Real *work,
00288              int *lwork,
00289              int *info);
00290 #else
00291
00320 void sgeqrf_(int *m,
00321              int *n,
00322              Real *a,
00323              int *lda,
00324              Real *tau,
00325              Real *work,
00326              int *lwork,
00327              int *info);
00328 #endif
00329
00330 #ifdef MTK_PRECISION_DOUBLE
00331
00365 void dormqr_(char *side,
00366              char *trans,
00367              int *m,
00368              int *n,
00369              int *k,
00370              Real *a,
00371              int *lda,
00372              Real *tau,
00373              Real *c,
00374              int *ldc,
00375              Real *work,
00376              int *lwork,
00377              int *info);
```

```

00378 #else
00379
00413 void sormqr_(char *side,
00414             char *trans,
00415             int *m,
00416             int *n,
00417             int *k,
00418             Real *a,
00419             int *lda,
00420             Real *tau,
00421             Real *c,
00422             int *ldc,
00423             Real *work,
00424             int *lwork,
00425             int *info);
00426 #endif
00427 }
00428 }
00429
00430 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
    mtk::Real *rhs) {
00431
00432
00433     #if MTK_DEBUG_LEVEL > 0
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     #if MTK_DEBUG_LEVEL > 0
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  #ifdef MTK_PRECISION_DOUBLE
00491  dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00492  #else
00493  fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00494  #endif
00495
00496  delete [] ipiv;
00497
00498  // After output, the data in the matrix will be column-major ordered.
00499
00500  bb.SetOrdering(mtk::COL_MAJOR);
00501
00502  #if MTK_DEBUG_LEVEL > 0
00503  std::cout << "bb_col_maj_ord =" << std::endl;
00504  std::cout << bb << std::endl;
00505  #endif
00506
00507  bb.OrderRowMajor();
00508
00509  #if MTK_DEBUG_LEVEL > 0
00510  std::cout << "bb_row_maj_ord =" << std::endl;
00511  std::cout << bb << std::endl;
00512  #endif
00513
00514  return info;
00515 }
00516
00517 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
                                mtk::UniStgGrid1D &rhs) {
00518
00519     int nrhs{1}; // Number of right-hand sides.
00520
00521     int *ipiv{}; // Array for pivoting information.
00522     int info{}; // Status of the solution.
00523     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00524
00525     try {
00526         ipiv = new int[mm_rank];
00527     } catch (std::bad_alloc &memory_allocation_exception) {
00528         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00529             std::endl;
00530         std::cerr << memory_allocation_exception.what() << std::endl;
00531     }
00532     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00533
00534     int lddb = mm_rank;
00535     int mm_ld = mm_rank;
00536
00537     mm.OrderColMajor();
00538
00539     #ifdef MTK_PRECISION_DOUBLE
00540     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00541         rhs.discrete_field_u(), &lddb, &info);
00542     #else
00543     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00544         rhs.discrete_field_u(), &lddb, &info);
00545     #endif
00546
00547     mm.OrderRowMajor();
00548
00549     delete [] ipiv;
00550
00551     return info;
00552 }
00553
00554 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
    (mtk::DenseMatrix &aa) {
00555
00556     mtk::Real *work{}; // Working array.
00557     mtk::Real *tau{}; // Array for the Householder scalars.
00558
00559     // Prepare to factorize: allocate and inquire for the value of lwork.
00560     try {
00561         work = new mtk::Real[1];
00562     } catch (std::bad_alloc &memory_allocation_exception) {
00563         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00564             std::endl;
00565         std::cerr << memory_allocation_exception.what() << std::endl;
00566     }
00567     memset(work, mtk::kZero, sizeof(aa.data()[0])*1);

```

```

00569
00570     int lwork{-1};
00571     int info{};
00572
00573     int aa_num_cols = aa.num_cols();
00574     int aaT_num_rows = aa.num_cols();
00575     int aaT_num_cols = aa.num_rows();
00576
00577     #if MTK_DEBUG_LEVEL > 0
00578     std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00579     std::cout << aa << std::endl;
00580     #endif
00581
00582     #ifdef MTK_PRECISION_DOUBLE
00583     dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00584            tau,
00585            work, &lwork, &info);
00586     #else
00587     fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00588            tau,
00589            work, &lwork, &info);
00590     #endif
00591
00592     #if MTK_DEBUG_LEVEL > 0
00593     if (info == 0) {
00594         lwork = (int) work[0];
00595     } else {
00596         std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00597             std::endl;
00598         std::cerr << "Exiting..." << std::endl;
00599     }
00600     #endif
00601
00602     #if MTK_DEBUG_LEVEL>0
00603     std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00604         << std::endl;
00605     #endif
00606
00607     delete [] work;
00608     work = nullptr;
00609
00610     // Once we know lwork, we can actually invoke the factorization:
00611     try {
00612         work = new mtk::Real [lwork];
00613     } catch (std::bad_alloc &memory_allocation_exception) {
00614         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00615             std::endl;
00616         std::cerr << memory_allocation_exception.what() << std::endl;
00617     }
00618     memset(work, mtk::kZero, sizeof(work[0])*lwork);
00619
00620     int ltau = std::min(aaT_num_rows, aaT_num_cols);
00621
00622     try {
00623         tau = new mtk::Real [ltau];
00624     } catch (std::bad_alloc &memory_allocation_exception) {
00625         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00626             std::endl;
00627         std::cerr << memory_allocation_exception.what() << std::endl;
00628     }
00629     memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00630
00631     #ifdef MTK_PRECISION_DOUBLE
00632     dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00633            tau, work, &lwork, &info);
00634     #else
00635     fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00636            tau, work, &lwork, &info);
00637     #endif
00638
00639     if (!info) {
00640         #if MTK_DEBUG_LEVEL > 0
00641         std::cout << "QR factorization completed!" << std::endl << std::endl;
00642         #endif
00643     } else {
00644         std::cerr << "Error solving system! info = " << info << std::endl;
00645         std::cerr << "Exiting..." << std::endl;
00646     }
00647
00648     #if MTK_DEBUG_LEVEL > 0
00649     std::cout << "Input matrix AFTER QR factorization:" << std::endl;

```

```

00650     std::cout << aa << std::endl;
00651     #endif
00652
00653     // We now generate the real matrix Q with orthonormal columns. This has to
00654     // be done separately since the actual output of dgegrf_ (AA_) represents
00655     // the orthogonal matrix Q as a product of min(aa_num_rows,aa_num_cols)
00656     // elementary Householder reflectors. Notice that we must re-inquire the new
00657     // value for lwork that is used.
00658
00659     bool padded{false};
00660
00661     bool transpose{false};
00662
00663     mtk::DenseMatrix QQ_(aa.num_cols(),padded,transpose);
00664
00665     #if MTK_DEBUG_LEVEL > 0
00666     std::cout << "Initialized QQ_T: " << std::endl;
00667     std::cout << QQ_ << std::endl;
00668     #endif
00669
00670     // Assemble the QQ_ matrix:
00671     lwork = -1;
00672
00673     delete[] work;
00674     work = nullptr;
00675
00676     try {
00677         work = new mtk::Real[lwork];
00678     } catch (std::bad_alloc &memory_allocation_exception) {
00679         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00680             std::endl;
00681         std::cerr << memory_allocation_exception.what() <<
00682             std::endl;
00683     }
00684     memset(work, mtk::kZero, sizeof(work[0])*lwork);
00685
00686     char side_{'L'};
00687     char trans_{'N'};
00688
00689     int aux = QQ_.num_rows();
00690
00691     #ifdef MTK_PRECISION_DOUBLE
00692     dormqr_(&side_, &trans_,
00693         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00694         QQ_.data(), &aux, work, &lwork, &info);
00695     #else
00696     formqr_(&side_, &trans_,
00697         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00698         QQ_.data(), &aux, work, &lwork, &info);
00699     #endif
00700
00701     #if MTK_DEBUG_LEVEL > 0
00702     if (info == 0) {
00703         lwork = (int) work[0];
00704     } else {
00705         std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00706         std::cerr << "Exiting..." << std::endl;
00707     }
00708     #endif
00709
00710     #if MTK_DEBUG_LEVEL > 0
00711     std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00712         std::endl << std::endl;
00713     #endif
00714
00715     delete[] work;
00716     work = nullptr;
00717
00718     try {
00719         work = new mtk::Real[lwork];
00720     } catch (std::bad_alloc &memory_allocation_exception) {
00721         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00722             std::endl;
00723         std::cerr << memory_allocation_exception.what() << std::endl;
00724     }
00725     memset(work, mtk::kZero, sizeof(work[0])*lwork);
00726
00727     #ifdef MTK_PRECISION_DOUBLE
00728     dormqr_(&side_, &trans_,
00729         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00730         QQ_.data(), &aux, work, &lwork, &info);

```

```

00731     #else
00732     formqr_(&side_, &trans_,
00733           &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00734           QQ_.data(), &aux, work, &lwork, &info);
00735     #endif
00736
00737     if (!info) {
00738         #if MTK_DEBUG_LEVEL>0
00739         std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00740         #endif
00741     } else {
00742         std::cerr << "Something went wrong solving system! info = " << info <<
00743         std::endl;
00744         std::cerr << "Exiting..." << std::endl;
00745     }
00746
00747     delete[] work;
00748     work = nullptr;
00749
00750     delete[] tau;
00751     tau = nullptr;
00752
00753     return QQ_;
00754 }
00755
00756 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
    mtk::DenseMatrix &aa,
00757
00758     mtk::Real *ob_,
00759     int ob_ld_) {
00760     // We first invoke the solver to query for the value of lwork. For this,
00761     // we must at least allocate enough space to allow access to WORK(1), or
00762     // work[0]:
00763
00764     // If LWORK = -1, then a workspace query is assumed; the routine only
00765     // calculates the optimal size of the WORK array, returns this value as
00766     // the first entry of the WORK array, and no error message related to
00767     // LWORK is issued by XERBLA.
00768
00769     mtk::Real *work{}; // Work array.
00770
00771     try {
00772         work = new mtk::Real[1];
00773     } catch (std::bad_alloc &memory_allocation_exception) {
00774         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00775         std::cerr << memory_allocation_exception.what() << std::endl;
00776     }
00777     memset(work, mtk::kZero, sizeof(work[0])*1);
00778
00779     char trans_{'N'};
00780     int nrhs_{1};
00781     int info{0};
00782     int lwork{-1};
00783
00784     int AA_num_rows_ = aa.num_rows();
00785     int AA_num_cols_ = aa.num_cols();
00786     int AA_ld_ = std::max(1, aa.num_cols());
00787
00788     #ifdef MTK_PRECISION_DOUBLE
00789     dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00790         ob_, &ob_ld_,
00791         work, &lwork, &info);
00792     #else
00793     sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00794         ob_, &ob_ld_,
00795         work, &lwork, &info);
00796     #endif
00797
00798     if (info == 0) {
00799         lwork = (int) work[0];
00800     } else {
00801         std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00802         std::endl;
00803         std::cerr << "Exiting..." << std::endl;
00804         return info;
00805     }
00806
00807     #if MTK_DEBUG_LEVEL > 0
00808     std::cout << "lwork = " << std::endl << std::setw(12)<< lwork <<
00809     std::endl << std::endl;
00810     #endif

```

```

00811
00812 // We then use lwork's new value to create the work array:
00813 delete[] work;
00814 work = nullptr;
00815
00816 try {
00817     work = new mtk::Real[lwork];
00818 } catch (std::bad_alloc &memory_allocation_exception) {
00819     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00820     std::cerr << memory_allocation_exception.what() << std::endl;
00821 }
00822 memset(work, 0.0, sizeof(work[0])*lwork);
00823
00824 // We now invoke the solver again:
00825 #ifdef MTK_PRECISION_DOUBLE
00826 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00827        ob_, &ob_ld_,
00828        work, &lwork, &info);
00829 #else
00830 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00831        ob_, &ob_ld_,
00832        work, &lwork, &info);
00833 #endif
00834
00835 delete [] work;
00836 work = nullptr;
00837
00838 return info;
00839 }

```

17.79 src/mtk_matrix.cc File Reference

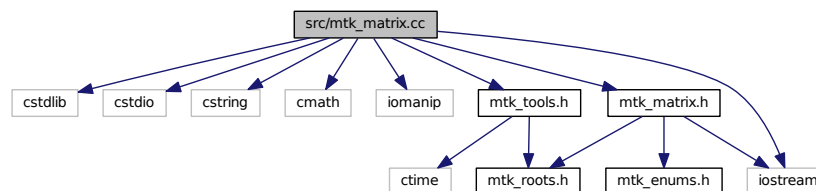
Implementing the representation of a matrix in the MTK.

```

#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:



17.79.1 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](#).

17.80 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::DENSE),
00069     ordering_(mtk::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() {}
00106
00107 mtk::MatrixStorage mtk::Matrix::storage() const {
00108
00109     return storage_;
00110 }
00111
00112 mtk::MatrixOrdering mtk::Matrix::ordering() const {
00113
00114     return ordering_;
00115 }
00116
00117 int mtk::Matrix::num_rows() const {
00118
00119     return num_rows_;
00120 }
00121
00122 int mtk::Matrix::num_cols() const {
00123
00124     return num_cols_;
00125 }
00126
00127 int mtk::Matrix::num_values() const {
00128
00129     return num_values_;
00130 }
00131
00132 int mtk::Matrix::ld() const {
00133
00134     return ld_;
00135 }
00136
00137 int mtk::Matrix::num_zero() const {
00138
00139     return num_zero_;
00140 }
00141
00142 int mtk::Matrix::num_non_zero() const {
00143
00144     return num_non_zero_;
00145 }
00146
00147 int mtk::Matrix::num_null() const {
00148
00149     return num_null_;
00150 }
00151
00152 int mtk::Matrix::num_non_null() const {
00153
00154     return num_non_null_;
00155 }
00156
00157 int mtk::Matrix::kl() const {

```

```

00158
00159     return kl_;
00160 }
00161
00162 int mtk::Matrix::ku() const {
00163
00164     return ku_;
00165 }
00166
00167 int mtk::Matrix::bandwidth() const {
00168
00169     return bandwidth_;
00170 }
00171
00172 mtk::Real mtk::Matrix::rel_density() const {
00173
00174     return rel_density_;
00175 }
00176
00177 mtk::Real mtk::Matrix::abs_sparsity() const {
00178
00179     return abs_sparsity_;
00180 }
00181
00182 mtk::Real mtk::Matrix::rel_sparsity() const {
00183
00184     return rel_sparsity_;
00185 }
00186
00187 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss) {
00188
00189     #if MTK_DEBUG_LEVEL > 0
00190     mtk::Tools::Prevent(!(ss == mtk::DENSE ||
00191                          ss == mtk::BANDED ||
00192                          ss == mtk::CRS),
00193                        __FILE__, __LINE__, __func__);
00194     #endif
00195
00196     storage_ = ss;
00197 }
00198
00199 void mtk::Matrix::set_ordering(const
00200 mtk::MatrixOrdering &oo) {
00201
00202     #if MTK_DEBUG_LEVEL > 0
00203     mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
00204                          mtk::COL_MAJOR),
00205                        __FILE__, __LINE__, __func__);
00206     #endif
00207
00208     ordering_ = oo;
00209
00210     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00211           std::max(1,num_cols_): std::max(1,num_rows_);
00212 }
00213
00214 void mtk::Matrix::set_num_rows(int in) {
00215
00216     #if MTK_DEBUG_LEVEL > 0
00217     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00218     #endif
00219
00220     num_rows_ = in;
00221     num_values_ = num_rows_*num_cols_;
00222     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00223           std::max(1,num_cols_): std::max(1,num_rows_);
00224 }
00225
00226 void mtk::Matrix::set_num_cols(int in) {
00227
00228     #if MTK_DEBUG_LEVEL > 0
00229     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00230     #endif
00231
00232     num_cols_ = in;
00233     num_values_ = num_rows_*num_cols_;
00234     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00235           std::max(1,num_cols_): std::max(1,num_rows_);
00236 }
00237
00238 void mtk::Matrix::set_num_zero(int in) {

```



```

00237
00238     #if MTK_DEBUG_LEVEL > 0
00239     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00240     #endif
00241
00242     num_zero_ = in;
00243     num_non_zero_ = num_values_ - num_zero_;
00244
00245     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00246     rel_sparsity_ = 1.0 - rel_density_;
00247 }
00248
00249
00250 void mtk::Matrix::set_num_null(int in) {
00251
00252     #if MTK_DEBUG_LEVEL > 0
00253     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00254     #endif
00255
00256     num_null_ = in;
00257     num_non_null_ = num_values_ - num_null_;
00258
00259     abs_density_ = (mtk::Real) num_non_null_/num_values_;
00260     abs_sparsity_ = 1.0 - abs_density_;
00261 }
00262
00263
00264 void mtk::Matrix::IncreaseNumZero() {
00265
00266     num_zero_++;
00267     num_non_zero_ = num_values_ - num_zero_;
00268     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00269     rel_sparsity_ = 1.0 - rel_density_;
00270 }
00271
00272
00273 void mtk::Matrix::IncreaseNumNull() {
00274
00275     num_null_++;
00276     num_non_null_ = num_values_ - num_null_;
00277     abs_density_ = (mtk::Real) num_non_null_/num_values_;
00278     abs_sparsity_ = 1.0 - abs_density_;
00279 }
00280
00281
00282 }

```

17.81 src/mtk_tools.cc File Reference

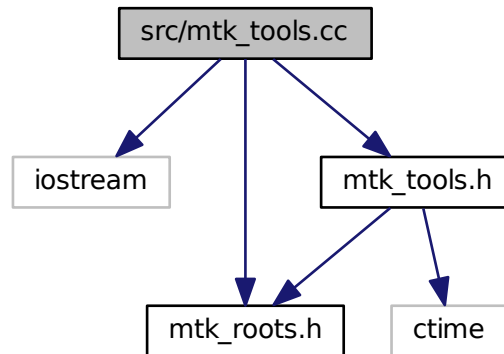
Implements a execution tool manager class.

```

#include <iostream>
#include "mtk_roots.h"
#include "mtk_tools.h"

```

Include dependency graph for mtk_tools.cc:



17.81.1 Detailed Description

Basic tools to ensure execution correctness.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_tools.cc](#).

17.82 mtk_tools.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
00058 #include "mtk_roots.h"
00059 #include "mtk_tools.h"
00060
00061 void mtk::Tools::Prevent(const bool condition,
00062                         const char *fname,
00063                         int lineno,
00064                         const char *fxname) {
00065
00066     #if MTK_DEBUG_LEVEL > 0
00067     if (lineno < 1) {
00070         std::cerr << __FILE__ << ": " << "Incorrect parameter at line " <<
00071         __LINE__ - 2 << " (" << __func__ << ")" << std::endl;
00072         exit(EXIT_FAILURE);
00073     }
00074     #endif
00075
00076     if (condition) {
00077         std::cerr << fname << ": " << "Incorrect parameter at line " <<
00078         lineno << " (" << fxname << ")" << std::endl;
00079         exit(EXIT_FAILURE);
00080     }
00081 }
00082
00083 int mtk::Tools::test_number_; // Used to control the correctness of the test.
00084
00085 mtk::Real mtk::Tools::duration_; // Duration of the current test.
00086
00087 clock_t mtk::Tools::begin_time_; // Used to time tests.
00088
00089 void mtk::Tools::BeginUnitTestNo(const int &nn) {
00090
00091     #if MTK_DEBUG_LEVEL > 0
00092     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00093     #endif
00094
00095     test_number_ = nn;
00096
00097     #if MTK_DEBUG_LEVEL > 0
00098     std::cout << "Beginning test " << nn << "." << std::endl;
00099     #endif
00100     begin_time_ = clock();
00101 }
00102
00103 void mtk::Tools::EndUnitTestNo(const int &nn) {
00104
00105     #if MTK_DEBUG_LEVEL > 0
00106     mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00107     #endif
00108
00109     duration_ = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00110 }
00111
00112 void mtk::Tools::Assert(const bool condition) {
00113
00114     if (condition) {
00115         std::cout << "Test " << test_number_ << ": PASSED in " << duration_ <<
00116         " s." << std::endl;
00117     }
00118 }

```

```

00119 } else {
00120     std::cout << "Test " << test_number_ << ": FAILED in " << duration_ <<
00121         " s." << std::endl;
00122 }
00123 }

```

17.83 src/mtk_uni_stg_grid_1d.cc File Reference

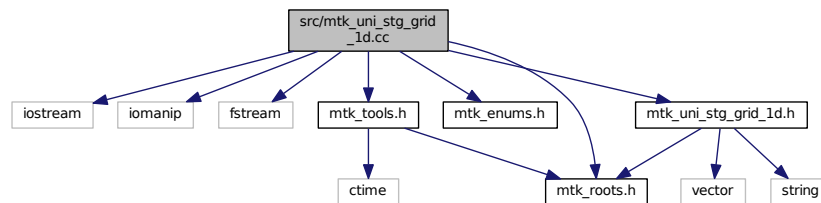
Implementation of an 1D uniform staggered grid.

```

#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)`

17.83.1 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](#).

17.84 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
00081
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_u_.size(); ++ii) {
00090         stream << std::setw(10) << in.discrete_field_u_[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_u_(),
00103     west_bndy_x_(),
00104     east_bndy_x_(),
00105     num_cells_x_(),
00106     delta_x_() {}
00107
00108 mtk::UniStgGrid1D::UniStgGrid1D(const
00109     UniStgGrid1D &grid):
00110     nature_(grid.nature_),
00111     west_bndy_x_(grid.west_bndy_x_),
00112     east_bndy_x_(grid.east_bndy_x_),
00113     num_cells_x_(grid.num_cells_x_),
00114     delta_x_(grid.delta_x_) {
00115
00116     std::copy(grid.discrete_domain_x_.begin(),
00117         grid.discrete_domain_x_.begin() + grid.
00118         discrete_domain_x_.size(),
00119         discrete_domain_x_.begin());
00120
00121     std::copy(grid.discrete_field_u_.begin(),
00122         grid.discrete_field_u_.begin() + grid.
00123         discrete_field_u_.size(),
00124         discrete_field_u_.begin());
00125 }
00126
00127 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00128     const Real &east_bndy_x,
00129     const int &num_cells_x,
00130     const mtk::FieldNature &nature) {
00131
00132     #if MTK_DEBUG_LEVEL > 0
00133     mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00134     mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00135     mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00136     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00137     #endif
00138
00139     nature_ = nature;
00140     west_bndy_x_ = west_bndy_x;
00141     east_bndy_x_ = east_bndy_x;
00142     num_cells_x_ = num_cells_x;
00143
00144     delta_x_ = (east_bndy_x - west_bndy_x) / ((mtk::Real) num_cells_x);
00145 }
00146
00147 mtk::UniStgGrid1D::~~UniStgGrid1D() {}
00148
00149 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00150
00151     return west_bndy_x_;
00152 }
00153
00154 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00155
00156     return east_bndy_x_;
00157 }
00158
00159 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00160
00161     return delta_x_;
00162 }
00163
00164 mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() {
00165
00166     return discrete_domain_x_.data();
00167 }
00168
00169 mtk::Real *mtk::UniStgGrid1D::discrete_field_u() {

```

```

00167
00168     return discrete_field_u_.data();
00169 }
00170
00171 int mtk::UniStgGrid1D::num_cells_x() const {
00172
00173     return num_cells_x_;
00174 }
00175
00176 void mtk::UniStgGrid1D::BindScalarField(
00177     mtk::Real (*ScalarField)(mtk::Real xx)) {
00178
00179     #if MTK_DEBUG_LEVEL > 0
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_u_.reserve(num_cells_x_ + 2);
00198
00199     discrete_field_u_.push_back(ScalarField(west_bndy_x_));
00200
00201     discrete_field_u_.push_back(ScalarField(first_center));
00202     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00203         discrete_field_u_.push_back(ScalarField(first_center + ii*delta_x_));
00204     }
00205     discrete_field_u_.push_back(ScalarField(east_bndy_x_));
00206 }
00207
00208 void mtk::UniStgGrid1D::BindVectorField(
00209     mtk::Real (*VectorField)(mtk::Real xx)) {
00210
00211     #if MTK_DEBUG_LEVEL > 0
00212     mtk::Tools::Prevent(nature_ == mtk::SCALAR, __FILE__, __LINE__, __func__);
00213     #endif
00214
00215     discrete_domain_x_.reserve(num_cells_x_ + 1);
00216
00217     discrete_domain_x_.push_back(west_bndy_x_);
00218     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00219         discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00220     }
00221     discrete_domain_x_.push_back(east_bndy_x_);
00222
00223     discrete_field_u_.reserve(num_cells_x_ + 1);
00224
00225     discrete_field_u_.push_back(VectorField(west_bndy_x_));
00226     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00227         discrete_field_u_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00228     }
00229     discrete_field_u_.push_back(VectorField(east_bndy_x_));
00230 }
00231
00232 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00233     std::string space_name,
00234     std::string field_name) {
00235
00236     std::ofstream output_dat_file; // Output file.
00237
00238     output_dat_file.open(filename);
00239
00240     if (!output_dat_file.is_open()) {
00241         return false;
00242     }
00243 }
00244
00245
00246
00247
00248
00249
00250
00251

```

```

00252     output_dat_file << "## " << space_name << ' ' << field_name << std::endl;
00253     for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00254         output_dat_file << discrete_domain_x_[ii] << ' ' << discrete_field_u_[ii] <<
00255             std::endl;
00256     }
00257
00258     output_dat_file.close();
00259
00260     return true;
00261 }

```

17.85 src/mtk_uni_stg_grid_2d.cc File Reference

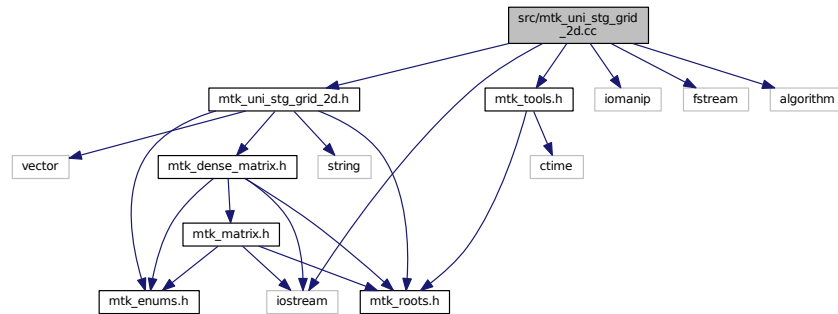
Implementation of a 2D uniform staggered grid.

```

#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)`

17.85.1 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](#).

17.86 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     stream << "x:";
00076
00077

```

```

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
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         int mm{in.num_cells_x_};
00103         int nn{in.num_cells_y_};
00104         int p_offset{nn*(mm + 1) - 1};
00105
00106         stream << "p(x,y):" << std::endl;
00107         for (int ii = 0; ii < nn; ++ii) {
00108             for (int jj = 0; jj < mm + 1; ++jj) {
00109                 stream << std::setw(10) << in.discrete_field_[ii*(mm + 1) + jj];
00110             }
00111             stream << std::endl;
00112         }
00113         stream << std::endl;
00114
00115         stream << "q(x,y):" << std::endl;
00116         for (int ii = 0; ii < nn + 1; ++ii) {
00117             for (int jj = 0; jj < mm; ++jj) {
00118                 stream << std::setw(10) <<
00119                     in.discrete_field_[p_offset + ii*mm + jj];
00120             }
00121             stream << std::endl;
00122         }
00123         stream << std::endl;
00124     }
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.

```

```

        discrete_domain_x_.size(),
        discrete_domain_x_.begin());
00158
00159
00160    std::copy(grid.discrete_domain_y_.begin(),
00161              grid.discrete_domain_y_.begin() + grid.
discrete_domain_y_.size(),
        discrete_domain_y_.begin());
00162
00163
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     #if MTK_DEBUG_LEVEL > 0
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
00189     nature_ = nature;
00190
00191     west_bndy_ = west_bndy;
00192     east_bndy_ = east_bndy;
00193     num_cells_x_ = num_cells_x;
00194
00195     south_bndy_ = south_bndy;
00196     north_bndy_ = north_bndy;
00197     num_cells_y_ = num_cells_y;
00198
00199     delta_x_ = (east_bndy_ - west_bndy_) / ((mtk::Real) num_cells_x);
00200     delta_y_ = (north_bndy_ - south_bndy_) / ((mtk::Real) num_cells_y);
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 mtk::Real mtk::UniStgGrid2D::south_bndy() const {
00231
00232     return south_bndy_;
00233 }
00234
00235 mtk::Real mtk::UniStgGrid2D::north_bndy() const {
00236

```

```

00237     return north_bndy_;
00238 }
00239
00240 int mtk::UniStgGrid2D::num_cells_y() const {
00241
00242     return num_cells_y_;
00243 }
00244
00245 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00246
00247     return delta_y_;
00248 }
00249
00250 void mtk::UniStgGrid2D::BindScalarField(Real (*ScalarField)(
    Real xx, Real yy)) {
00251
00252     #if MTK_DEBUG_LEVEL > 0
00253     mtk::Tools::Prevent(nature_ != mtk::SCALAR, __FILE__, __LINE__, __func__);
00254     #endif
00255
00256
00257     discrete_domain_x_.reserve(num_cells_x_ + 2);
00258
00259     discrete_domain_x_.push_back(west_bndy_);
00260     #ifdef MTK_PRECISION_DOUBLE
00261     auto first_center = west_bndy_ + delta_x_/2.0;
00262     #else
00263     auto first_center = west_bndy_ + delta_x_/2.0f;
00264     #endif
00265     discrete_domain_x_.push_back(first_center);
00266     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00267         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00268     }
00269     discrete_domain_x_.push_back(east_bndy_);
00270
00271
00272     discrete_domain_y_.reserve(num_cells_y_ + 2);
00273
00274     discrete_domain_y_.push_back(south_bndy_);
00275     #ifdef MTK_PRECISION_DOUBLE
00276     first_center = south_bndy_ + delta_x_/2.0;
00277     #else
00278     first_center = south_bndy_ + delta_x_/2.0f;
00279     #endif
00280     discrete_domain_y_.push_back(first_center);
00281     for (auto ii = 1; ii < num_cells_y_; ++ii) {
00282         discrete_domain_y_.push_back(first_center + ii*delta_y_);
00283     }
00284     discrete_domain_y_.push_back(north_bndy_);
00285
00286
00287     discrete_field_.reserve((num_cells_x_ + 2)*(num_cells_y_ + 2));
00288
00289     for (int ii = 0; ii < num_cells_x_ + 2; ++ii) {
00290         for (int jj = 0; jj < num_cells_y_ + 2; ++jj) {
00291             discrete_field_.push_back(ScalarField(discrete_domain_x_[ii],
00292                                                     discrete_domain_y_[jj]));
00293         }
00294     }
00295 }
00296
00297 void mtk::UniStgGrid2D::BindVectorFieldPComponent(
00298     mtk::Real (*VectorField)(mtk::Real xx, mtk::Real yy)) {
00299
00300     int mm{num_cells_x_};
00301     int nn{num_cells_y_};
00302
00303     int total{nn*(mm + 1) + mm*(nn + 1)};
00304
00305     #ifdef MTK_PRECISION_DOUBLE
00306     double half_delta_x{delta_x_/2.0};
00307     double half_delta_y{delta_y_/2.0};
00308     #else
00309     float half_delta_x{delta_x_/2.0f};
00310     float half_delta_y{delta_y_/2.0f};
00311     #endif
00312
00313     // We need every data point of the discrete domain; i.e. we need all the
00314     // nodes and all the centers. There are mm centers for the x direction, and
00315     // nn centers for the y direction. Since there is one node per center, that

```

```

00321 // amounts to 2*mm. If we finally consider the final boundary node, it
00322 // amounts to a total of 2*mm + 1 for the x direction. Analogously, for the
00323 // y direction, this amounts to 2*nn + 1.
00324
00325 discrete_domain_x_.reserve(2*mm + 1);
00326
00327 discrete_domain_x_.push_back(west_bndy_);
00328 for (int ii = 1; ii < (2*mm + 1); ++ii) {
00329     discrete_domain_x_.push_back(west_bndy_ + ii*half_delta_x);
00330 }
00331
00332
00333 discrete_domain_y_.reserve(2*nn + 1);
00334
00335 discrete_domain_y_.push_back(south_bndy_);
00336 for (int ii = 1; ii < (2*nn + 1); ++ii) {
00337     discrete_domain_y_.push_back(south_bndy_ + ii*half_delta_y);
00338 }
00339
00340
00341 discrete_field_.reserve(total);
00342
00343 // For each y-center.
00344 for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00345
00346     // Bind all of the x-nodes for this y-center.
00347     for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00348         discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00349                                                 discrete_domain_y_[ii]));
00350
00351         #if MTK_DEBUG_LEVEL > 0
00352         std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00353             discrete_domain_y_[ii] << " = " <<
00354             VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00355         #endif
00356     }
00357 }
00358
00359 #if MTK_DEBUG_LEVEL > 0
00360 std::cout << std::endl;
00361 #endif
00362 }
00363
00364 void mtk::UniStgGrid2D::BindVectorFieldQComponent(
00365     mtk::Real (*VectorField)(mtk::Real xx, mtk::Real yy)) {
00366
00367     int mm{num_cells_x_};
00368     int nn{num_cells_y_};
00369
00370
00371 // For each y-node.
00372 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00373
00374     // Bind all of the x-center for this y-node.
00375     for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00376         discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00377                                                 discrete_domain_y_[ii]));
00378
00379         #if MTK_DEBUG_LEVEL > 0
00380         std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00381             discrete_domain_y_[ii] << " = " <<
00382             VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00383         #endif
00384     }
00385 }
00386
00387 #if MTK_DEBUG_LEVEL > 0
00388 std::cout << std::endl;
00389 #endif
00390 }
00391
00392 void mtk::UniStgGrid2D::BindVectorField(
00393     Real (*VectorFieldPComponent)(Real xx,Real yy),
00394     Real (*VectorFieldQComponent)(Real xx,Real yy)) {
00395
00396     #if MTK_DEBUG_LEVEL > 0
00397     mtk::Tools::Prevent(nature_ != mtk::VECTOR, __FILE__, __LINE__, __func__);
00398     #endif
00399
00400     BindVectorFieldPComponent(VectorFieldPComponent);
00401     BindVectorFieldQComponent(VectorFieldQComponent);
00402 }
00403
00404

```

```

00405 bool mtk::UniStgGrid2D::WriteToFile(std::string filename,
00406                                     std::string space_name_x,
00407                                     std::string space_name_y,
00408                                     std::string field_name) {
00409
00410     std::ofstream output_dat_file; // Output file.
00411
00412     output_dat_file.open(filename);
00413
00414     if (!output_dat_file.is_open()) {
00415         return false;
00416     }
00417
00418     if (nature_ == mtk::SCALAR) {
00419         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00420             field_name << std::endl;
00421
00422         for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00423             for (unsigned int jj = 0; jj < discrete_domain_y_.size(); ++jj) {
00424                 output_dat_file << discrete_domain_x_[ii] << ' ' <<
00425                     discrete_domain_y_[jj] << ' ' <<
00426                     discrete_field_[ii*discrete_domain_y_.size() + jj] <<
00427                     std::endl;
00428             }
00429             output_dat_file << std::endl;
00430         }
00431     } else {
00432         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00433             field_name << std::endl;
00434
00435         output_dat_file << "# Horizontal component:" << std::endl;
00436
00437         int mm{num_cells_x_};
00438         int nn{num_cells_y_};
00439
00440         // For each y-center.
00441         int idx{};
00442         for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00443             // Bind all of the x-nodes for this y-center.
00444             for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00445
00446                 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00447                     discrete_domain_y_[ii] << ' ' << discrete_field_[idx] << ' ' <<
00448                     mtk::kZero << std::endl;
00449
00450                 ++idx;
00451             }
00452         }
00453
00454         int p_offset{nn*(mm + 1) - 1};
00455         idx = 0;
00456         output_dat_file << "# Vertical component:" << std::endl;
00457         // For each y-node.
00458         for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00459             // Bind all of the x-center for this y-node.
00460             for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00461
00462                 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00463                     discrete_domain_y_[ii] << ' ' << mtk::kZero << ' ' <<
00464                     discrete_field_[p_offset + idx] << std::endl;
00465
00466                 ++idx;
00467             }
00468         }
00469     }
00470
00471     output_dat_file.close();
00472
00473     return true;
00474 }

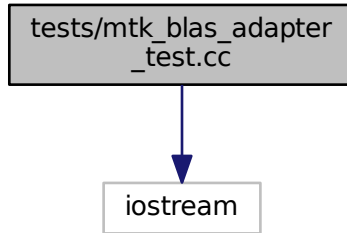
```

17.87 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](#) ()

17.87.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_blas_adapter_test.cc](#).

17.87.2 Function Documentation

17.87.2.1 int main ()

Definition at line [109](#) of file [mtk_blas_adapter_test.cc](#).

17.88 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

```



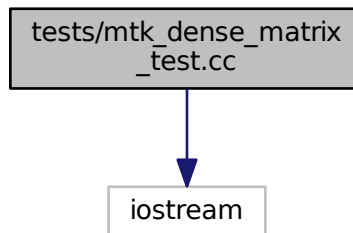
```
00105 #else
00106 #include <iostream>
00107 using std::cout;
00108 using std::endl;
00109 int main () {
00110     cout << "This code HAS to be compiled with support for C++11." << endl;
00111     cout << "Exiting..." << endl;
00112 }
00113 #endif
```

17.89 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 ()`

17.89.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_dense_matrix_test.cc](#).

17.89.2 Function Documentation

17.89.2.1 `int main ()`

Definition at line [330](#) of file [mtk_dense_matrix_test.cc](#).

17.90 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 int main () {
00312
00313     std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00314
00315     TestDefaultConstructor();
00316     TestConstructorWithNumRowsNumCols();
00317     TestConstructAsIdentity();
00318     TestConstructAsVandermonde();
00319     TestSetValueGetValue();
00320     TestConstructAsVandermondeTranspose();
00321     TestKron();
00322     TestConstructWithNumRowsNumColsAssignmentOperator();
00323     TestConstructAsVandermondeTransposeAssignmentOperator();
00324 }
00325
00326 #else
00327 #include <iostream>

```

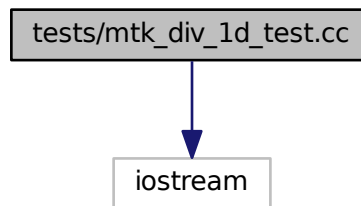
```
00328 using std::cout;
00329 using std::endl;
00330 int main () {
00331     cout << "This code HAS to be compiled with support for C++11." << endl;
00332     cout << "Exiting..." << endl;
00333 }
00334 #endif
```

17.91 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 ()`

17.91.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_div_1d_test.cc](#).

17.91.2 Function Documentation

17.91.2.1 `int main ()`

Definition at line [288](#) of file [mtk_div_1d_test.cc](#).

17.92 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
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062     mtk::Tools::BeginUnitTestNo(1);
00063
00064     mtk::Div1D div2;
00065
00066     bool assertion = div2.ConstructDiv1D();
00067
00068     if (!assertion) {
00069         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00070     }
00071
00072     mtk::Tools::EndUnitTestNo(1);
00073     mtk::Tools::Assert(assertion);
00074 }
00075
00076 void TestDefaultConstructorFactoryMethodFourthOrder() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Div1D div4;
00081
00082     bool assertion = div4.ConstructDiv1D(4);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00086     }
00087
00088     mtk::Tools::EndUnitTestNo(2);

```

```
00089     mtk::Tools::Assert(assertion);
00090 }
00091
00092 void TestDefaultConstructorFactoryMethodSixthOrder() {
00093
00094     mtk::Tools::BeginUnitTestNo(3);
00095
00096     mtk::Div1D div6;
00097
00098     bool assertion = div6.ConstructDiv1D(6);
00099
00100     if (!assertion) {
00101         std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00102     }
00103
00104     mtk::Tools::EndUnitTestNo(3);
00105     mtk::Tools::Assert(assertion);
00106 }
00107
00108 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00109
00110     mtk::Tools::BeginUnitTestNo(4);
00111
00112     mtk::Div1D div8;
00113
00114     bool assertion = div8.ConstructDiv1D(8);
00115
00116     if (!assertion) {
00117         std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00118     }
00119
00120     mtk::Tools::EndUnitTestNo(4);
00121     mtk::Tools::Assert(assertion);
00122 }
00123
00124 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00125
00126     mtk::Tools::BeginUnitTestNo(5);
00127
00128     mtk::Div1D div10;
00129
00130     bool assertion = div10.ConstructDiv1D(10);
00131
00132     if (!assertion) {
00133         std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00134     }
00135
00136     mtk::Tools::EndUnitTestNo(5);
00137     mtk::Tools::Assert(assertion);
00138 }
00139
00140 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00141
00142     mtk::Tools::BeginUnitTestNo(6);
00143
00144     mtk::Div1D div12;
00145
00146     bool assertion = div12.ConstructDiv1D(12);
00147
00148     if (!assertion) {
00149         std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00150     }
00151
00152     mtk::Tools::EndUnitTestNo(6);
00153     mtk::Tools::Assert(assertion);
00154 }
00155
00156 void TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold() {
00157
00158     mtk::Tools::BeginUnitTestNo(7);
00159
00160     mtk::Div1D div14;
00161
00162     bool assertion = div14.ConstructDiv1D(14);
00163
00164     if (!assertion) {
00165         std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00166     }
00167
00168     mtk::Tools::EndUnitTestNo(7);
00169     mtk::Tools::Assert(assertion);
```



```

00170 }
00171
00172 void TestSecondOrderReturnAsDenseMatrixWithGrid() {
00173     mtk::Tools::BeginUnitTestNo(8);
00174
00175     mtk::Div1D div2;
00176
00177     bool assertion = div2.ConstructDiv1D();
00178
00179     if (!assertion) {
00180         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00181     }
00182
00183     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00184
00185     mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00186
00187     int rr{7};
00188     int cc{6};
00189
00190     mtk::DenseMatrix ref(rr, cc);
00191
00192     // Row 2.
00193     ref.SetValue(1,0,-5.0);
00194     ref.SetValue(1,1,5.0);
00195     ref.SetValue(1,2,0.0);
00196     ref.SetValue(1,3,0.0);
00197     ref.SetValue(1,4,0.0);
00198     ref.SetValue(1,5,0.0);
00199     ref.SetValue(1,6,0.0);
00200
00201     // Row 3.
00202     ref.SetValue(2,0,0.0);
00203     ref.SetValue(2,1,-5.0);
00204     ref.SetValue(2,2,5.0);
00205     ref.SetValue(2,3,0.0);
00206     ref.SetValue(2,4,0.0);
00207     ref.SetValue(2,5,0.0);
00208     ref.SetValue(2,6,0.0);
00209
00210     // Row 4.
00211     ref.SetValue(3,0,0.0);
00212     ref.SetValue(3,1,0.0);
00213     ref.SetValue(3,2,-5.0);
00214     ref.SetValue(3,3,5.0);
00215     ref.SetValue(3,4,0.0);
00216     ref.SetValue(3,5,0.0);
00217     ref.SetValue(3,6,0.0);
00218
00219     // Row 5.
00220     ref.SetValue(4,0,0.0);
00221     ref.SetValue(4,1,0.0);
00222     ref.SetValue(4,2,0.0);
00223     ref.SetValue(4,3,-5.0);
00224     ref.SetValue(4,4,5.0);
00225     ref.SetValue(4,5,0.0);
00226     ref.SetValue(4,6,0.0);
00227
00228     // Row 6.
00229     ref.SetValue(5,0,0.0);
00230     ref.SetValue(5,1,0.0);
00231     ref.SetValue(5,2,0.0);
00232     ref.SetValue(5,3,0.0);
00233     ref.SetValue(5,4,-5.0);
00234     ref.SetValue(5,5,5.0);
00235     ref.SetValue(5,6,0.0);
00236
00237     assertion = assertion && (div2m == ref);
00238
00239     mtk::Tools::EndUnitTestNo(8);
00240     mtk::Tools::Assert(assertion);
00241 }
00242
00243 void TestFourthOrderReturnAsDenseMatrixWithGrid() {
00244
00245     mtk::Tools::BeginUnitTestNo(9);
00246
00247     mtk::Div1D div4;
00248
00249     bool assertion = div4.ConstructDiv1D(4);
00250

```

```

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

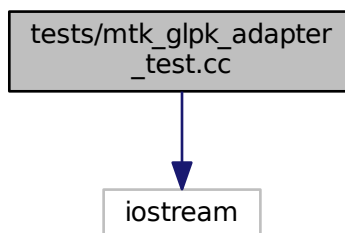
```

17.93 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 ()`

17.93.1 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`.

17.93.2 Function Documentation

17.93.2.1 `int main ()`

Definition at line 81 of file `mtk_glpk_adapter_test.cc`.

17.94 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

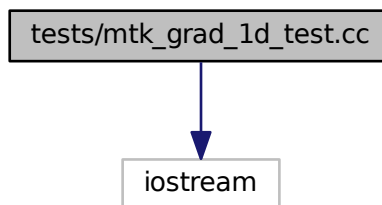
```

17.95 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` ()

17.95.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_1d_test.cc](#).

17.95.2 Function Documentation**17.95.2.1 int main ()**

Definition at line 296 of file [mtk_grad_1d_test.cc](#).

17.96 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
(10));
00243
00244     std::cout << grad4m << std::endl;
00245
00246     mtk::Tools::EndUnitTestNo(7);
00247     mtk::Tools::Assert(assertion);
00248 }
00249
00250 void TestWriteToFile() {
00251
00252     mtk::Tools::BeginUnitTestNo(8);
00253
00254     mtk::Grad1D grad2;
00255
00256     bool assertion = grad2.ConstructGrad1D();
00257
00258     if (!assertion) {
00259         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00260     }
00261
00262     mtk::UniStgGrid1D grid(0.0, 1.0, 50);
00263
00264     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00265
00266     std::cout << grad2m << std::endl;
00267
00268     assertion = assertion && grad2m.WriteToFile("mtk_grad_1d_test_08.dat");
00269
00270     if (!assertion) {
00271         std::cerr << "Error writing to file." << std::endl;
00272     }
00273
00274     mtk::Tools::EndUnitTestNo(8);
00275     mtk::Tools::Assert(assertion);
00276 }
00277
00278 int main () {
00279
00280     std::cout << "Testing mtk::Grad1D class." << std::endl;
00281
00282     TestDefaultConstructorFactoryMethodDefault();
00283     TestDefaultConstructorFactoryMethodFourthOrder();
00284     TestDefaultConstructorFactoryMethodSixthOrder();
00285     TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00286     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00287     TestReturnAsDenseMatrixWithGrid();
00288     TestReturnAsDimensionlessDenseMatrix();
00289     TestWriteToFile();
00290 }
00291
00292 #else
00293 #include <iostream>
00294 using std::cout;
00295 using std::endl;
00296 int main () {
00297     cout << "This code HAS to be compiled with support for C++11." << endl;
00298     cout << "Exiting..." << endl;
00299 }
00300 #endif

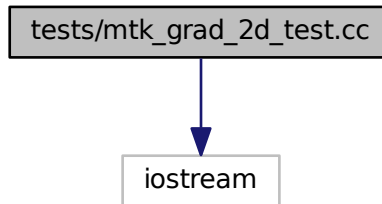
```


17.97 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 ()`

17.97.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_2d_test.cc](#).

17.97.2 Function Documentation

17.97.2.1 `int main ()`

Definition at line [139](#) of file [mtk_grad_2d_test.cc](#).

17.98 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

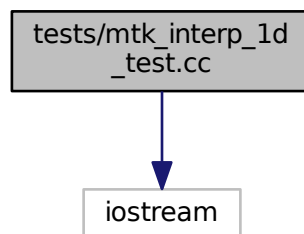
```

17.99 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()`

17.99.1 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](#).

17.99.2 Function Documentation

17.99.2.1 `int main()`

Definition at line 113 of file [mtk_interp_1d_test.cc](#).

17.100 `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>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061     mtk::Tools::BeginUnitTestNo(1);
00062     mtk::InterplD inter;
00063     bool assertion = inter.ConstructInterplD();
00064     if (!assertion) {
00065         std::cerr << "Mimetic interp could not be built." << std::endl;
00066     }
00067     mtk::Tools::EndUnitTestNo(1);
00068     mtk::Tools::Assert(assertion);
00069 }
00070
00071 void TestReturnAsDenseMatrixWithGrid() {
00072     mtk::Tools::BeginUnitTestNo(2);
00073     mtk::InterplD inter;
00074     bool assertion = inter.ConstructInterplD();
00075     if (!assertion) {
00076         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00077     }
00078     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00079     mtk::DenseMatrix interpm(inter.ReturnAsDenseMatrix(grid));
00080     assertion =
00081         assertion && interpm.GetValue(0,0) == 1.0 && interpm.GetValue(5,6) == 1.0;
00082     mtk::Tools::EndUnitTestNo(2);
00083     mtk::Tools::Assert(assertion);
00084 }
00085
00086 int main () {
00087     std::cout << "Testing mtk::InterplD class." << std::endl;
00088     TestDefaultConstructorFactoryMethodDefault();
00089     TestReturnAsDenseMatrixWithGrid();
00090 }
00091 #else
00092 #include <iostream>
00093 using std::cout;
00094 using std::endl;
00095 int main () {
00096     cout << "This code HAS to be compiled with support for C++11." << endl;
00097     cout << "Exiting..." << endl;
00098 }
00099 #endif

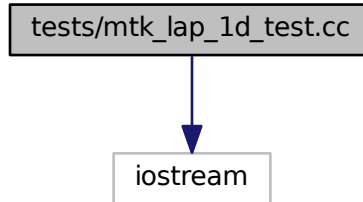
```

17.101 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](#) ()

17.101.1 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_lap_1d_test.cc](#).

17.101.2 Function Documentation

17.101.2.1 int main ()

Definition at line [193](#) of file [mtk_lap_1d_test.cc](#).

17.102 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();

```



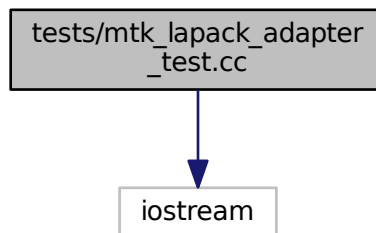
```
00188     TestReturnAsDenseMatrix();
00189 }
00190
00191 #else
00192 #include <iostream>
00193 int main () {
00194     std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00195     std::cout << "Exiting..." << std::endl;
00196 }
00197 #endif
```

17.103 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](#) ()

17.103.1 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](#).

17.103.2 Function Documentation

17.103.2.1 int main ()

Definition at line [81](#) of file [mtk_lapack_adapter_test.cc](#).

17.104 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

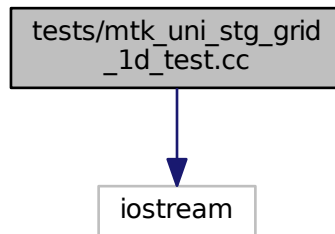
```

17.105 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](#) ()

17.105.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_1d_test.cc](#).

17.105.2 Function Documentation

17.105.2.1 int main ()

Definition at line [172](#) of file [mtk_uni_stg_grid_1d_test.cc](#).

17.106 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(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_u()[0] == 0.0 &&
00113         gg.discrete_field_u()[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_u()[0] == 0.0 &&
00146         gg.discrete_field_u()[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

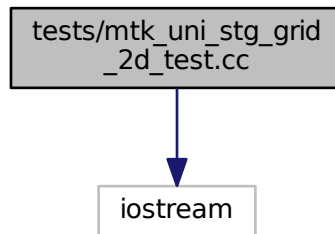
```

17.107 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](#) ()

17.107.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_2d_test.cc](#).

17.107.2 Function Documentation

17.107.2.1 `int main ()`

Definition at line [202](#) of file [mtk_uni_stg_grid_2d_test.cc](#).

17.108 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(mtk::Real xx, 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(mtk::Real xx, mtk::Real yy) {
00153
00154     return xx + 0.01;
00155 }
00156
00157 mtk::Real VectorFieldQComponent(mtk::Real xx, 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
00189     std::cout << "Testing mtk::UniStgGrid2D class." << std::endl;
00190
00191     TestDefaultConstructor();
00192     TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator();
00193     TestGetters();
00194     TestBindScalarFieldWriteToFile();
00195     TestBindVectorField();
00196 }
00197
00198 #else
00199 #include <iostream>
00200 using std::cout;
00201 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
```

Index

BANDED
Enumerations., [34](#)

COL_MAJOR
Enumerations., [34](#)

CRS
Enumerations., [34](#)

DENSE
Enumerations., [34](#)

Data structures., [36](#)

Enumerations., [33](#)
BANDED, [34](#)
COL_MAJOR, [34](#)
CRS, [34](#)
DENSE, [34](#)
ROW_MAJOR, [34](#)
SCALAR, [33](#)
SCALAR_TO_VECTOR, [33](#)
VECTOR, [33](#)
VECTOR_TO_SCALAR, [33](#)

Execution tools., [35](#)

Grids., [38](#)

Mimetic operators., [39](#)

mtk, [41](#)
operator<<, [43](#), [44](#)

Numerical methods., [37](#)

operator<<
mtk, [43](#), [44](#)

ROW_MAJOR
Enumerations., [34](#)

Real
Roots., [32](#)

Roots., [31](#)
Real, [32](#)

SCALAR
Enumerations., [33](#)

SCALAR_TO_VECTOR
Enumerations., [33](#)

VECTOR
Enumerations., [33](#)

VECTOR_TO_SCALAR
Enumerations., [33](#)