

## MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.6

Thu Oct 15 2015 13:15:34



# Contents

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

<b>14 Module Documentation</b>	<b>31</b>
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
<b>15 Namespace Documentation</b>	<b>41</b>
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	<a href="#">operator&lt;&lt;</a>	43
15.1.1.5	<a href="#">operator&lt;&lt;</a>	44
15.1.1.6	<a href="#">operator&lt;&lt;</a>	44
15.1.1.7	<a href="#">operator&lt;&lt;</a>	44
15.1.1.8	<a href="#">saxpy_</a>	45
15.1.1.9	<a href="#">sgels_</a>	45
15.1.1.10	<a href="#">sgemm_</a>	46
15.1.1.11	<a href="#">sgemv_</a>	47
15.1.1.12	<a href="#">sgeqrf_</a>	47
15.1.1.13	<a href="#">sgesv_</a>	47
15.1.1.14	<a href="#">snrm2_</a>	48
15.1.1.15	<a href="#">sormqr_</a>	48
<b>16</b>	<b>Class Documentation</b>	<b>51</b>
16.1	<a href="#">mtk::BCDesc1D Class Reference</a>	51
16.1.1	<a href="#">Detailed Description</a>	51
16.1.2	<a href="#">Member Function Documentation</a>	51
16.1.2.1	<a href="#">ImposeOnGrid</a>	52
16.1.2.2	<a href="#">ImposeOnOperator</a>	52
16.2	<a href="#">mtk::BLASAdapter Class Reference</a>	53
16.2.1	<a href="#">Detailed Description</a>	53
16.2.2	<a href="#">Member Function Documentation</a>	54
16.2.2.1	<a href="#">RealAXPY</a>	54
16.2.2.2	<a href="#">RealDenseMM</a>	54
16.2.2.3	<a href="#">RealDenseMV</a>	57
16.2.2.4	<a href="#">RealNRM2</a>	58
16.2.2.5	<a href="#">RelNorm2Error</a>	59
16.3	<a href="#">mtk::DenseMatrix Class Reference</a>	59
16.3.1	<a href="#">Detailed Description</a>	62
16.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	62
16.3.2.1	<a href="#">DenseMatrix</a>	62
16.3.2.2	<a href="#">DenseMatrix</a>	62
16.3.2.3	<a href="#">DenseMatrix</a>	63
16.3.2.4	<a href="#">DenseMatrix</a>	64
16.3.2.5	<a href="#">DenseMatrix</a>	64
16.3.2.6	<a href="#">~DenseMatrix</a>	65
16.3.3	<a href="#">Member Function Documentation</a>	65

16.3.3.1	<a href="#">data</a>	65
16.3.3.2	<a href="#">GetValue</a>	66
16.3.3.3	<a href="#">Kron</a>	67
16.3.3.4	<a href="#">matrix_properties</a>	68
16.3.3.5	<a href="#">num_cols</a>	69
16.3.3.6	<a href="#">num_rows</a>	69
16.3.3.7	<a href="#">operator=</a>	70
16.3.3.8	<a href="#">OrderColMajor</a>	71
16.3.3.9	<a href="#">OrderRowMajor</a>	72
16.3.3.10	<a href="#">SetOrdering</a>	72
16.3.3.11	<a href="#">SetValue</a>	73
16.3.3.12	<a href="#">Transpose</a>	74
16.3.4	<a href="#">Friends And Related Function Documentation</a>	74
16.3.4.1	<a href="#">operator&lt;&lt;</a>	75
16.3.5	<a href="#">Member Data Documentation</a>	75
16.3.5.1	<a href="#">data_</a>	75
16.3.5.2	<a href="#">matrix_properties_</a>	75
16.4	<a href="#">mtk::Div1D Class Reference</a>	75
16.4.1	<a href="#">Detailed Description</a>	78
16.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	78
16.4.2.1	<a href="#">Div1D</a>	78
16.4.2.2	<a href="#">Div1D</a>	78
16.4.2.3	<a href="#">~Div1D</a>	78
16.4.3	<a href="#">Member Function Documentation</a>	79
16.4.3.1	<a href="#">AssembleOperator</a>	79
16.4.3.2	<a href="#">coeffs_interior</a>	79
16.4.3.3	<a href="#">ComputePreliminaryApproximations</a>	79
16.4.3.4	<a href="#">ComputeRationalBasisNullSpace</a>	80
16.4.3.5	<a href="#">ComputeStencilBoundaryGrid</a>	81
16.4.3.6	<a href="#">ComputeStencilInteriorGrid</a>	81
16.4.3.7	<a href="#">ComputeWeights</a>	82
16.4.3.8	<a href="#">ConstructDiv1D</a>	83
16.4.3.9	<a href="#">mim_bndy</a>	83
16.4.3.10	<a href="#">num_bndy_coeffs</a>	84
16.4.3.11	<a href="#">ReturnAsDenseMatrix</a>	84
16.4.3.12	<a href="#">weights_cbs</a>	85
16.4.3.13	<a href="#">weights_crs</a>	85

16.4.4	Friends And Related Function Documentation	85
16.4.4.1	operator<<	85
16.4.5	Member Data Documentation	85
16.4.5.1	coeffs_interior_	85
16.4.5.2	dim_null_	86
16.4.5.3	divergence_	86
16.4.5.4	divergence_length_	86
16.4.5.5	mim_bndy_	86
16.4.5.6	mimetic_threshold_	86
16.4.5.7	minrow_	86
16.4.5.8	num_bndy_coeffs_	86
16.4.5.9	order_accuracy_	86
16.4.5.10	prem_apps_	86
16.4.5.11	rat_basis_null_space_	86
16.4.5.12	row_	86
16.4.5.13	weights_cbs_	87
16.4.5.14	weights_crs_	87
16.5	mtk::Div2D Class Reference	87
16.5.1	Detailed Description	89
16.5.2	Constructor & Destructor Documentation	89
16.5.2.1	Div2D	89
16.5.2.2	Div2D	89
16.5.2.3	~Div2D	89
16.5.3	Member Function Documentation	89
16.5.3.1	ConstructDiv2D	89
16.5.3.2	ReturnAsDenseMatrix	90
16.5.4	Member Data Documentation	90
16.5.4.1	divergence_	90
16.5.4.2	mimetic_threshold_	90
16.5.4.3	order_accuracy_	91
16.6	mtk::GLPKAdapter Class Reference	91
16.6.1	Detailed Description	91
16.6.2	Member Function Documentation	92
16.6.2.1	SolveSimplexAndCompare	92
16.7	mtk::Grad1D Class Reference	93
16.7.1	Detailed Description	96
16.7.2	Constructor & Destructor Documentation	96

16.7.2.1	Grad1D	96
16.7.2.2	Grad1D	96
16.7.2.3	~Grad1D	97
16.7.3	Member Function Documentation	97
16.7.3.1	AssembleOperator	97
16.7.3.2	coeffs_interior	97
16.7.3.3	ComputePreliminaryApproximations	97
16.7.3.4	ComputeRationalBasisNullSpace	98
16.7.3.5	ComputeStencilBoundaryGrid	99
16.7.3.6	ComputeStencilInteriorGrid	99
16.7.3.7	ComputeWeights	100
16.7.3.8	ConstructGrad1D	100
16.7.3.9	mim_bndy	101
16.7.3.10	num_bndy_coeffs	102
16.7.3.11	ReturnAsDenseMatrix	102
16.7.3.12	ReturnAsDenseMatrix	103
16.7.3.13	weights_cbs	103
16.7.3.14	weights_crs	104
16.7.4	Friends And Related Function Documentation	104
16.7.4.1	operator<<	104
16.7.5	Member Data Documentation	104
16.7.5.1	coeffs_interior_	104
16.7.5.2	dim_null_	104
16.7.5.3	gradient_	104
16.7.5.4	gradient_length_	104
16.7.5.5	mim_bndy_	104
16.7.5.6	mimetic_threshold_	104
16.7.5.7	minrow_	105
16.7.5.8	num_bndy_approxs_	105
16.7.5.9	num_bndy_coeffs_	105
16.7.5.10	order_accuracy_	105
16.7.5.11	prem_apps_	105
16.7.5.12	rat_basis_null_space_	105
16.7.5.13	row_	105
16.7.5.14	weights_cbs_	105
16.7.5.15	weights_crs_	105
16.8	mtk::Grad2D Class Reference	105



16.8.1	Detailed Description	107
16.8.2	Constructor & Destructor Documentation	107
16.8.2.1	Grad2D	107
16.8.2.2	Grad2D	107
16.8.2.3	~Grad2D	107
16.8.3	Member Function Documentation	107
16.8.3.1	ConstructGrad2D	107
16.8.3.2	ReturnAsDenseMatrix	108
16.8.4	Member Data Documentation	108
16.8.4.1	gradient_	108
16.8.4.2	mimetic_threshold_	108
16.8.4.3	order_accuracy_	108
16.9	mtk::Interp1D Class Reference	109
16.9.1	Detailed Description	110
16.9.2	Constructor & Destructor Documentation	110
16.9.2.1	Interp1D	110
16.9.2.2	Interp1D	110
16.9.2.3	~Interp1D	110
16.9.3	Member Function Documentation	110
16.9.3.1	coeffs_interior	110
16.9.3.2	ConstructInterp1D	111
16.9.3.3	ReturnAsDenseMatrix	111
16.9.4	Friends And Related Function Documentation	111
16.9.4.1	operator<<	112
16.9.5	Member Data Documentation	112
16.9.5.1	coeffs_interior_	112
16.9.5.2	dir_interp_	112
16.9.5.3	order_accuracy_	112
16.10	mtk::Interp2D Class Reference	112
16.10.1	Detailed Description	114
16.10.2	Constructor & Destructor Documentation	114
16.10.2.1	Interp2D	114
16.10.2.2	Interp2D	114
16.10.2.3	~Interp2D	114
16.10.3	Member Function Documentation	114
16.10.3.1	ConstructInterp2D	114
16.10.3.2	ReturnAsDenseMatrix	114

16.10.4 Member Data Documentation . . . . .	114
16.10.4.1 interpolator_ . . . . .	114
16.10.4.2 mimetic_threshold_ . . . . .	114
16.10.4.3 order_accuracy_ . . . . .	114
16.11 mtk::Lap1D Class Reference . . . . .	115
16.11.1 Detailed Description . . . . .	116
16.11.2 Constructor & Destructor Documentation . . . . .	116
16.11.2.1 Lap1D . . . . .	116
16.11.2.2 Lap1D . . . . .	116
16.11.2.3 ~Lap1D . . . . .	116
16.11.3 Member Function Documentation . . . . .	116
16.11.3.1 ConstructLap1D . . . . .	116
16.11.3.2 Data . . . . .	117
16.11.3.3 ReturnAsDenseMatrix . . . . .	118
16.11.4 Friends And Related Function Documentation . . . . .	119
16.11.4.1 operator<< . . . . .	119
16.11.5 Member Data Documentation . . . . .	119
16.11.5.1 laplacian_ . . . . .	119
16.11.5.2 laplacian_length_ . . . . .	119
16.11.5.3 mimetic_threshold_ . . . . .	119
16.11.5.4 order_accuracy_ . . . . .	119
16.12 mtk::Lap2D Class Reference . . . . .	119
16.12.1 Detailed Description . . . . .	121
16.12.2 Constructor & Destructor Documentation . . . . .	121
16.12.2.1 Lap2D . . . . .	121
16.12.2.2 Lap2D . . . . .	121
16.12.2.3 ~Lap2D . . . . .	121
16.12.3 Member Function Documentation . . . . .	121
16.12.3.1 ConstructLap2D . . . . .	121
16.12.3.2 ReturnAsDenseMatrix . . . . .	121
16.12.4 Member Data Documentation . . . . .	121
16.12.4.1 laplacian_ . . . . .	121
16.12.4.2 mimetic_threshold_ . . . . .	121
16.12.4.3 order_accuracy_ . . . . .	122
16.13 mtk::LAPACKAdapter Class Reference . . . . .	122
16.13.1 Detailed Description . . . . .	122
16.13.2 Member Function Documentation . . . . .	123

16.13.2.1 QRFactorDenseMatrix . . . . .	123
16.13.2.2 SolveDenseSystem . . . . .	124
16.13.2.3 SolveDenseSystem . . . . .	125
16.13.2.4 SolveDenseSystem . . . . .	126
16.13.2.5 SolveRectangularDenseSystem . . . . .	126
16.14mtk::Matrix Class Reference . . . . .	127
16.14.1 Detailed Description . . . . .	130
16.14.2 Constructor & Destructor Documentation . . . . .	130
16.14.2.1 Matrix . . . . .	130
16.14.2.2 Matrix . . . . .	131
16.14.2.3 ~Matrix . . . . .	132
16.14.3 Member Function Documentation . . . . .	132
16.14.3.1 abs_density . . . . .	132
16.14.3.2 abs_sparsity . . . . .	132
16.14.3.3 bandwidth . . . . .	132
16.14.3.4 IncreaseNumNull . . . . .	132
16.14.3.5 IncreaseNumZero . . . . .	133
16.14.3.6 kl . . . . .	133
16.14.3.7 ku . . . . .	133
16.14.3.8 ld . . . . .	133
16.14.3.9 num_cols . . . . .	133
16.14.3.10num_non_null . . . . .	134
16.14.3.11num_non_zero . . . . .	134
16.14.3.12num_null . . . . .	134
16.14.3.13num_rows . . . . .	134
16.14.3.14num_values . . . . .	135
16.14.3.15num_zero . . . . .	135
16.14.3.16ordering . . . . .	135
16.14.3.17rel_density . . . . .	136
16.14.3.18rel_sparsity . . . . .	136
16.14.3.19set_num_cols . . . . .	136
16.14.3.20set_num_null . . . . .	137
16.14.3.21set_num_rows . . . . .	138
16.14.3.22set_num_zero . . . . .	138
16.14.3.23set_ordering . . . . .	139
16.14.3.24set_storage . . . . .	140
16.14.3.25storage . . . . .	140

16.14.4 Member Data Documentation . . . . .	141
16.14.4.1 abs_density_ . . . . .	141
16.14.4.2 abs_sparsity_ . . . . .	141
16.14.4.3 bandwidth_ . . . . .	141
16.14.4.4 kl_ . . . . .	141
16.14.4.5 ku_ . . . . .	141
16.14.4.6 ld_ . . . . .	141
16.14.4.7 num_cols_ . . . . .	141
16.14.4.8 num_non_null_ . . . . .	142
16.14.4.9 num_non_zero_ . . . . .	142
16.14.4.10 num_null_ . . . . .	142
16.14.4.11 num_rows_ . . . . .	142
16.14.4.12 num_values_ . . . . .	142
16.14.4.13 num_zero_ . . . . .	142
16.14.4.14 ordering_ . . . . .	142
16.14.4.15 rel_density_ . . . . .	142
16.14.4.16 rel_sparsity_ . . . . .	142
16.14.4.17 storage_ . . . . .	142
16.15 mtk::Quad1D Class Reference . . . . .	143
16.15.1 Detailed Description . . . . .	144
16.15.2 Constructor & Destructor Documentation . . . . .	144
16.15.2.1 Quad1D . . . . .	144
16.15.2.2 Quad1D . . . . .	144
16.15.2.3 ~Quad1D . . . . .	145
16.15.3 Member Function Documentation . . . . .	145
16.15.3.1 degree_approximation . . . . .	145
16.15.3.2 Integrate . . . . .	145
16.15.3.3 weights . . . . .	145
16.15.4 Friends And Related Function Documentation . . . . .	145
16.15.4.1 operator<< . . . . .	145
16.15.5 Member Data Documentation . . . . .	145
16.15.5.1 degree_approximation_ . . . . .	145
16.15.5.2 weights_ . . . . .	145
16.16 mtk::Tools Class Reference . . . . .	145
16.16.1 Detailed Description . . . . .	146
16.16.2 Member Function Documentation . . . . .	146
16.16.2.1 BeginTestNo . . . . .	146

16.16.2.2 EndTestNo . . . . .	147
16.16.2.3 Prevent . . . . .	147
16.16.3 Member Data Documentation . . . . .	148
16.16.3.1 begin_time_ . . . . .	149
16.16.3.2 test_number_ . . . . .	149
16.17mtk::UniStgGrid1D Class Reference . . . . .	149
16.17.1 Detailed Description . . . . .	152
16.17.2 Constructor & Destructor Documentation . . . . .	152
16.17.2.1 UniStgGrid1D . . . . .	152
16.17.2.2 UniStgGrid1D . . . . .	152
16.17.2.3 UniStgGrid1D . . . . .	152
16.17.2.4 ~UniStgGrid1D . . . . .	152
16.17.3 Member Function Documentation . . . . .	153
16.17.3.1 BindScalarField . . . . .	153
16.17.3.2 BindVectorField . . . . .	153
16.17.3.3 delta_x . . . . .	154
16.17.3.4 discrete_domain_x . . . . .	154
16.17.3.5 discrete_field_u . . . . .	154
16.17.3.6 east_bndy_x . . . . .	155
16.17.3.7 num_cells_x . . . . .	155
16.17.3.8 west_bndy_x . . . . .	155
16.17.3.9 WriteToFile . . . . .	155
16.17.4 Friends And Related Function Documentation . . . . .	156
16.17.4.1 operator<< . . . . .	156
16.17.5 Member Data Documentation . . . . .	156
16.17.5.1 delta_x_ . . . . .	156
16.17.5.2 discrete_domain_x_ . . . . .	156
16.17.5.3 discrete_field_u_ . . . . .	156
16.17.5.4 east_bndy_x_ . . . . .	156
16.17.5.5 nature_ . . . . .	156
16.17.5.6 num_cells_x_ . . . . .	156
16.17.5.7 west_bndy_x_ . . . . .	157
16.18mtk::UniStgGrid2D Class Reference . . . . .	157
16.18.1 Detailed Description . . . . .	160
16.18.2 Constructor & Destructor Documentation . . . . .	160
16.18.2.1 UniStgGrid2D . . . . .	160
16.18.2.2 UniStgGrid2D . . . . .	160

16.18.2.3	UniStgGrid2D	160
16.18.2.4	~UniStgGrid2D	161
16.18.3	Member Function Documentation	161
16.18.3.1	BindScalarField	161
16.18.3.2	BindVectorFieldPComponent	161
16.18.3.3	BindVectorFieldQComponent	162
16.18.3.4	delta_x	162
16.18.3.5	delta_y	162
16.18.3.6	discrete_domain_x	162
16.18.3.7	discrete_domain_y	162
16.18.3.8	discrete_field_u	162
16.18.3.9	east_bndy_x	163
16.18.3.10	north_bndy_y	163
16.18.3.11	num_cells_x	163
16.18.3.12	num_cells_y	163
16.18.3.13	south_bndy_y	164
16.18.3.14	west_bndy_x	164
16.18.3.15	WriteToFile	165
16.18.4	Friends And Related Function Documentation	165
16.18.4.1	operator<<	165
16.18.5	Member Data Documentation	165
16.18.5.1	delta_x_	165
16.18.5.2	delta_y_	166
16.18.5.3	discrete_domain_x_	166
16.18.5.4	discrete_domain_y_	166
16.18.5.5	discrete_field_u_	166
16.18.5.6	east_bndy_x_	166
16.18.5.7	nature_	166
16.18.5.8	north_bndy_y_	166
16.18.5.9	num_cells_x_	166
16.18.5.10	num_cells_y_	166
16.18.5.11	south_bndy_y_	166
16.18.5.12	west_bndy_x_	166
<b>17</b>	<b>File Documentation</b>	<b>169</b>
17.1	examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference	169
17.1.1	Detailed Description	169

17.1.2	Function Documentation	170
17.1.2.1	main	170
17.2	minimalistic_poisson_1d.cc	170
17.3	examples/poisson_1d/poisson_1d.cc File Reference	172
17.3.1	Detailed Description	173
17.3.2	Function Documentation	173
17.3.2.1	main	173
17.4	poisson_1d.cc	173
17.5	include/mtk.h File Reference	176
17.5.1	Detailed Description	177
17.6	mtk.h	177
17.7	include/mtk_bc_desc_1d.h File Reference	178
17.8	mtk_bc_desc_1d.h	179
17.9	include/mtk_blas_adapter.h File Reference	180
17.9.1	Detailed Description	181
17.10	mtk_blas_adapter.h	181
17.11	include/mtk_dense_matrix.h File Reference	182
17.11.1	Detailed Description	183
17.12	mtk_dense_matrix.h	184
17.13	include/mtk_div_1d.h File Reference	185
17.13.1	Detailed Description	186
17.14	mtk_div_1d.h	187
17.15	include/mtk_div_2d.h File Reference	188
17.15.1	Detailed Description	190
17.16	mtk_div_2d.h	190
17.17	include/mtk_enums.h File Reference	191
17.17.1	Detailed Description	191
17.18	mtk_enums.h	192
17.19	include/mtk_glpk_adapter.h File Reference	193
17.19.1	Detailed Description	194
17.20	mtk_glpk_adapter.h	194
17.21	include/mtk_grad_1d.h File Reference	195
17.21.1	Detailed Description	196
17.22	mtk_grad_1d.h	196
17.23	include/mtk_grad_2d.h File Reference	198
17.23.1	Detailed Description	199
17.24	mtk_grad_2d.h	199

17.25include/mtk_interp_1d.h File Reference . . . . .	200
17.25.1 Detailed Description . . . . .	202
17.26mtk_interp_1d.h . . . . .	202
17.27include/mtk_interp_2d.h File Reference . . . . .	203
17.27.1 Detailed Description . . . . .	204
17.28mtk_interp_2d.h . . . . .	205
17.29include/mtk_lap_1d.h File Reference . . . . .	206
17.29.1 Detailed Description . . . . .	207
17.30mtk_lap_1d.h . . . . .	207
17.31include/mtk_lap_2d.h File Reference . . . . .	208
17.31.1 Detailed Description . . . . .	210
17.32mtk_lap_2d.h . . . . .	210
17.33include/mtk_lapack_adapter.h File Reference . . . . .	211
17.33.1 Detailed Description . . . . .	212
17.34mtk_lapack_adapter.h . . . . .	212
17.35include/mtk_matrix.h File Reference . . . . .	213
17.35.1 Detailed Description . . . . .	214
17.36mtk_matrix.h . . . . .	214
17.37include/mtk_quad_1d.h File Reference . . . . .	216
17.37.1 Detailed Description . . . . .	217
17.38mtk_quad_1d.h . . . . .	218
17.39include/mtk_roots.h File Reference . . . . .	219
17.39.1 Detailed Description . . . . .	220
17.40mtk_roots.h . . . . .	220
17.41include/mtk_tools.h File Reference . . . . .	221
17.41.1 Detailed Description . . . . .	222
17.42mtk_tools.h . . . . .	222
17.43include/mtk_uni_stg_grid_1d.h File Reference . . . . .	223
17.43.1 Detailed Description . . . . .	224
17.44mtk_uni_stg_grid_1d.h . . . . .	224
17.45include/mtk_uni_stg_grid_2d.h File Reference . . . . .	226
17.45.1 Detailed Description . . . . .	227
17.46mtk_uni_stg_grid_2d.h . . . . .	227
17.47Makefile.inc File Reference . . . . .	228
17.48Makefile.inc . . . . .	229
17.49README.md File Reference . . . . .	231
17.50README.md . . . . .	231



17.51src/mtk_bc_desc_1d.cc File Reference . . . . .	233
17.52mtk_bc_desc_1d.cc . . . . .	233
17.53src/mtk_blas_adapter.cc File Reference . . . . .	234
17.54mtk_blas_adapter.cc . . . . .	235
17.55src/mtk_dense_matrix.cc File Reference . . . . .	238
17.55.1 Detailed Description . . . . .	239
17.56mtk_dense_matrix.cc . . . . .	239
17.57src/mtk_div_1d.cc File Reference . . . . .	246
17.57.1 Detailed Description . . . . .	246
17.58mtk_div_1d.cc . . . . .	247
17.59src/mtk_div_2d.cc File Reference . . . . .	263
17.59.1 Detailed Description . . . . .	264
17.60mtk_div_2d.cc . . . . .	264
17.61src/mtk_glpk_adapter.cc File Reference . . . . .	266
17.61.1 Detailed Description . . . . .	266
17.62mtk_glpk_adapter.cc . . . . .	267
17.63src/mtk_grad_1d.cc File Reference . . . . .	271
17.63.1 Detailed Description . . . . .	271
17.64mtk_grad_1d.cc . . . . .	272
17.65src/mtk_grad_2d.cc File Reference . . . . .	290
17.65.1 Detailed Description . . . . .	290
17.66mtk_grad_2d.cc . . . . .	291
17.67src/mtk_interp_1d.cc File Reference . . . . .	292
17.67.1 Detailed Description . . . . .	293
17.68mtk_interp_1d.cc . . . . .	293
17.69src/mtk_lap_1d.cc File Reference . . . . .	295
17.69.1 Detailed Description . . . . .	296
17.70mtk_lap_1d.cc . . . . .	296
17.71src/mtk_lap_2d.cc File Reference . . . . .	300
17.72mtk_lap_2d.cc . . . . .	301
17.73src/mtk_lapack_adapter.cc File Reference . . . . .	302
17.73.1 Detailed Description . . . . .	303
17.74mtk_lapack_adapter.cc . . . . .	303
17.75src/mtk_matrix.cc File Reference . . . . .	311
17.75.1 Detailed Description . . . . .	311
17.76mtk_matrix.cc . . . . .	311
17.77src/mtk_tools.cc File Reference . . . . .	315

17.77.1 Detailed Description . . . . .	315
17.78mtk_tools.cc . . . . .	315
17.79src/mtk_uni_stg_grid_1d.cc File Reference . . . . .	317
17.79.1 Detailed Description . . . . .	317
17.80mtk_uni_stg_grid_1d.cc . . . . .	318
17.81src/mtk_uni_stg_grid_2d.cc File Reference . . . . .	321
17.81.1 Detailed Description . . . . .	322
17.82mtk_uni_stg_grid_2d.cc . . . . .	322
17.83tests/mtk_blas_adapter_test.cc File Reference . . . . .	324
17.83.1 Detailed Description . . . . .	325
17.83.2 Function Documentation . . . . .	325
17.83.2.1 main . . . . .	325
17.84mtk_blas_adapter_test.cc . . . . .	325
17.85tests/mtk_dense_matrix_test.cc File Reference . . . . .	327
17.85.1 Detailed Description . . . . .	327
17.85.2 Function Documentation . . . . .	327
17.85.2.1 main . . . . .	327
17.86mtk_dense_matrix_test.cc . . . . .	327
17.87tests/mtk_div_1d_test.cc File Reference . . . . .	331
17.87.1 Detailed Description . . . . .	332
17.87.2 Function Documentation . . . . .	332
17.87.2.1 main . . . . .	332
17.88mtk_div_1d_test.cc . . . . .	332
17.89tests/mtk_glpk_adapter_test.cc File Reference . . . . .	335
17.89.1 Detailed Description . . . . .	335
17.89.2 Function Documentation . . . . .	336
17.89.2.1 main . . . . .	336
17.90mtk_glpk_adapter_test.cc . . . . .	336
17.91tests/mtk_grad_1d_test.cc File Reference . . . . .	337
17.91.1 Detailed Description . . . . .	337
17.91.2 Function Documentation . . . . .	338
17.91.2.1 main . . . . .	338
17.92mtk_grad_1d_test.cc . . . . .	338
17.93tests/mtk_interp_1d_test.cc File Reference . . . . .	340
17.93.1 Detailed Description . . . . .	341
17.93.2 Function Documentation . . . . .	341
17.93.2.1 main . . . . .	341

17.94	mtk_interp_1d_test.cc . . . . .	341
17.95	tests/mtk_lap_1d_test.cc File Reference . . . . .	343
17.95.1	Function Documentation . . . . .	343
17.95.1.1	main . . . . .	343
17.96	mtk_lap_1d_test.cc . . . . .	343
17.97	tests/mtk_lapack_adapter_test.cc File Reference . . . . .	345
17.97.1	Detailed Description . . . . .	346
17.97.2	Function Documentation . . . . .	346
17.97.2.1	main . . . . .	346
17.98	mtk_lapack_adapter_test.cc . . . . .	346
17.99	tests/mtk_uni_stg_grid_1d_test.cc File Reference . . . . .	347
17.99.1	Detailed Description . . . . .	348
17.99.2	Function Documentation . . . . .	348
17.99.2.1	main . . . . .	348
17.100	mtk_uni_stg_grid_1d_test.cc . . . . .	348
17.101	tests/mtk_uni_stg_grid_2d_test.cc File Reference . . . . .	350
17.101.1	Detailed Description . . . . .	351
17.101.2	Function Documentation . . . . .	351
17.101.2.1	main . . . . .	351
17.102	mtk_uni_stg_grid_2d_test.cc . . . . .	351
<b>Index</b>		<b>354</b>



# 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 a set of classes for **mimetic quadratures**, **mimetic interpolation**, and **mimetic discretization methods** for the numerical solution of ordinary and partial differential equations.

### 1.1 MTK Concerns

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

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

In order of dependence these are:

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

### 1.2 MTK 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 - 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.

### 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.





## 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](mailto:esanchez@mail.sdsu.edu) and a copy of the modified files should be reported once modifications are completed. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
4. 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.
5. Usage of the binary form on proprietary applications shall require explicit prior written permission from the the copyright holders.
6. 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 discretization** methods for the numerical solution of ordinary and partial differential equations.

#### 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. BLAS - Available from: <http://www.netlib.org/blas/>
  2. LAPACK - Available from: <http://www.netlib.org/lapack/>
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:
```

---

- make: builds only the library and the examples.
- all: builds the library, the examples and the documentation.
- mtklib: builds the library, i.e. generates the archive files.
- test: generates the tests.
- example: generates the examples.
- 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 processor 1.73GHz 2048 KB of cache and stepping of 8  
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)

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 arrays have to be created.

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

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

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

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

**Class `mtk::GLPKAdapter`**

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

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

Review the definition of sparse matrices properties.

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

Review the definition of sparse matrices properties.

**Member `mtk::Tools::Prevent` (`const bool condition, 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_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 mechanism.

**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:

<a href="#">mtk</a>	Mimetic Methods Toolkit namespace . . . . .	<a href="#">41</a>
---------------------	---	--------------------



## Chapter 12

# Class Index

### 12.1 Class List

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

<a href="#">mtk::BCDesc1D</a>	51
<a href="#">mtk::BLASAdapter</a>	
Adapter class for the BLAS API	53
<a href="#">mtk::DenseMatrix</a>	
Defines a common dense matrix, using a 1D array	59
<a href="#">mtk::Div1D</a>	
Implements a 1D mimetic divergence operator	75
<a href="#">mtk::Div2D</a>	87
<a href="#">mtk::GLPKAdapter</a>	
Adapter class for the GLPK API	91
<a href="#">mtk::Grad1D</a>	
Implements a 1D mimetic gradient operator	93
<a href="#">mtk::Grad2D</a>	105
<a href="#">mtk::Interp1D</a>	
Implements a 1D interpolation operator	109
<a href="#">mtk::Interp2D</a>	112
<a href="#">mtk::Lap1D</a>	
Implements a 1D mimetic Laplacian operator	115
<a href="#">mtk::Lap2D</a>	119
<a href="#">mtk::LAPACKAdapter</a>	
Adapter class for the LAPACK API	122
<a href="#">mtk::Matrix</a>	
Definition of the representation of a matrix in the MTK	127
<a href="#">mtk::Quad1D</a>	
Implements a 1D mimetic quadrature	143
<a href="#">mtk::Tools</a>	
Tool manager class	145
<a href="#">mtk::UniStgGrid1D</a>	
Uniform 1D Staggered Grid	149
<a href="#">mtk::UniStgGrid2D</a>	
Uniform 2D Staggered Grid	157



## Chapter 13

# File Index

### 13.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Makefile.inc</a>	228
examples/minimalistic_poisson_1d/ <a href="#">minimalistic_poisson_1d.cc</a>	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	170
examples/poisson_1d/ <a href="#">poisson_1d.cc</a>	
Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	173
include/ <a href="#">mtk.h</a>	
Includes the entire API	177
include/ <a href="#">mtk_bc_desc_1d.h</a>	179
include/ <a href="#">mtk_blas_adapter.h</a>	
Adapter class for the BLAS API	181
include/ <a href="#">mtk_dense_matrix.h</a>	
Defines a common dense matrix, using a 1D array	184
include/ <a href="#">mtk_div_1d.h</a>	
Includes the definition of the class Div1D	187
include/ <a href="#">mtk_div_2d.h</a>	
Includes the definition of the class Div2D	190
include/ <a href="#">mtk_enums.h</a>	
Considered enumeration types in the MTK	192
include/ <a href="#">mtk_glpk_adapter.h</a>	
Adapter class for the GLPK API	194
include/ <a href="#">mtk_grad_1d.h</a>	
Includes the definition of the class Grad1D	196
include/ <a href="#">mtk_grad_2d.h</a>	
Includes the definition of the class Grad2D	199
include/ <a href="#">mtk_interp_1d.h</a>	
Includes the definition of the class Interp1D	202
include/ <a href="#">mtk_interp_2d.h</a>	
Includes the definition of the class Interp2D	205
include/ <a href="#">mtk_lap_1d.h</a>	
Includes the definition of the class Lap1D	207
include/ <a href="#">mtk_lap_2d.h</a>	
Includes the implementation of the class Lap2D	210
include/ <a href="#">mtk_lapack_adapter.h</a>	
Adapter class for the LAPACK API	212

include/mtk_matrix.h	Definition of the representation of a matrix in the MTK	214
include/mtk_quad_1d.h	Includes the definition of the class Quad1D	218
include/mtk_roots.h	Fundamental definitions to be used across all classes of the MTK	220
include/mtk_tools.h	Tool manager class	222
include/mtk_uni_stg_grid_1d.h	Definition of an 1D uniform staggered grid	224
include/mtk_uni_stg_grid_2d.h	Definition of an 2D uniform staggered grid	227
src/mtk_bc_desc_1d.cc		233
src/mtk_blas_adapter.cc		235
src/mtk_dense_matrix.cc	Implements a common dense matrix, using a 1D array	239
src/mtk_div_1d.cc	Implements the class Div1D	247
src/mtk_div_2d.cc	Implements the class Div2D	264
src/mtk_glpk_adapter.cc	Adapter class for the GLPK API	267
src/mtk_grad_1d.cc	Implements the class Grad1D	272
src/mtk_grad_2d.cc	Implements the class Grad2D	291
src/mtk_interp_1d.cc	Includes the implementation of the class Interp1D	293
src/mtk_lap_1d.cc	Includes the implementation of the class Lap1D	296
src/mtk_lap_2d.cc		301
src/mtk_lapack_adapter.cc	Adapter class for the LAPACK API	303
src/mtk_matrix.cc	Implementing the representation of a matrix in the MTK	311
src/mtk_tools.cc	Implements a execution tool manager class	315
src/mtk_uni_stg_grid_1d.cc	Implementation of an 1D uniform staggered grid	318
src/mtk_uni_stg_grid_2d.cc	Implementation of a 2D uniform staggered grid	322
tests/mtk_blas_adapter_test.cc	Test file for the <code>mtk::BLASAdapter</code> class	325
tests/mtk_dense_matrix_test.cc	Test file for the <code>mtk::DenseMatrix</code> class	327
tests/mtk_div_1d_test.cc	Testing the mimetic 1D divergence, constructed with the CBS algorithm	332
tests/mtk_glpk_adapter_test.cc	Test file for the <code>mtk::GLPKAdapter</code> class	336
tests/mtk_grad_1d_test.cc	Testing the mimetic 1D gradient, constructed with the CBS algorithm	338
tests/mtk_interp_1d_test.cc	Testing the 1D interpolation	341
tests/mtk_lap_1d_test.cc		343



tests/ <a href="#">mtk_lapack_adapter_test.cc</a>	
Test file for the <a href="#">mtk::LAPACKAdapter</a> class . . . . .	346
tests/ <a href="#">mtk_uni_stg_grid_1d_test.cc</a>	
Test file for the <a href="#">mtk::UniStgGrid1D</a> class . . . . .	348
tests/ <a href="#">mtk_uni_stg_grid_2d_test.cc</a>	
Test file for the <a href="#">mtk::UniStgGrid2D</a> class . . . . .	351



## 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`

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

Enumerator

**`SCALAR`** Scalar-valued field.

**`VECTOR`** Vector-valued field.

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

#### 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](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 [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 66 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 )`

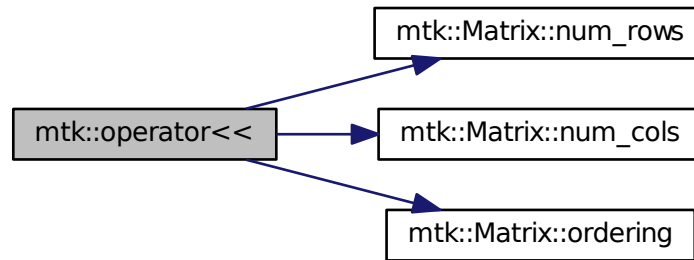
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

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 75 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 )`

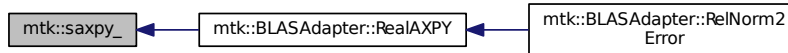
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

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

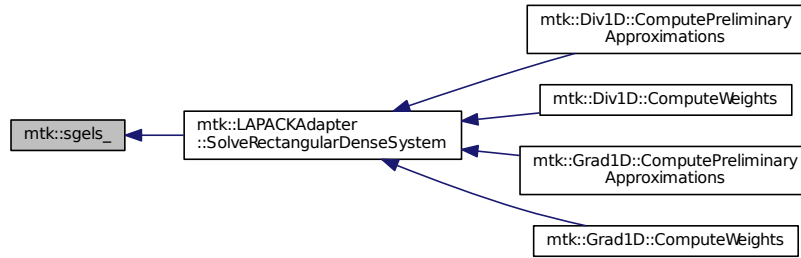
<http://www.math.utah.edu/software/lapack/lapack-s/sgels.html>

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

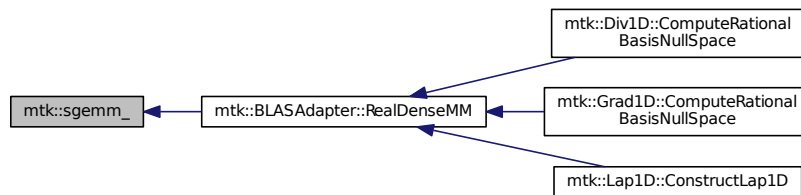
in, out	<i>b</i>	On entry, matrix b of right-hand side vectors.
in	<i>ldb</i>	The leading dimension of b. $ldb \geq \max(1, m, n)$ .
in, out	<i>work</i>	On exit, if info = 0, work(1) is optimal lwork.
in, out	<i>lwork</i>	The dimension of the array work.
in, out	<i>info</i>	If info = 0, then successful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

Here is the caller graph for this function:



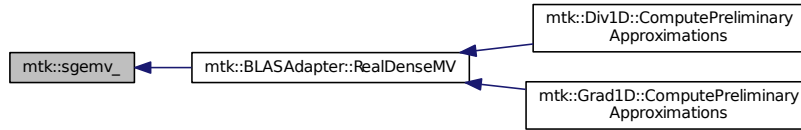
15.1.1.10 `void mtk::sgemm_ ( char * transa, char * transb, int * m, int * n, int * k, double * alpha, double * a, int * lda, double * b, aamm int * ldb, double * beta, double * c, int * ldc )`

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

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

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

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

See Also

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

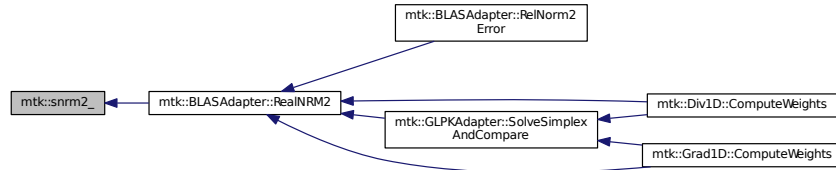
#### Parameters

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



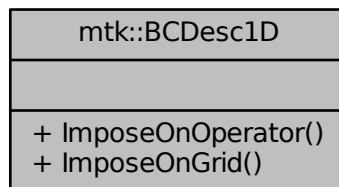
## Chapter 16

# 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 `ImposeOnOperator` (`DenseMatrix` &matrix, const std::vector< `Real` > &west, const std::vector< `Real` > &east)
- static void `ImposeOnGrid` (`UniStgGrid1D` &grid, const `Real` &omega, const `Real` &epsilon)

#### 16.1.1 Detailed Description

Definition at line 9 of file `mtk_bc_desc_1d.h`.

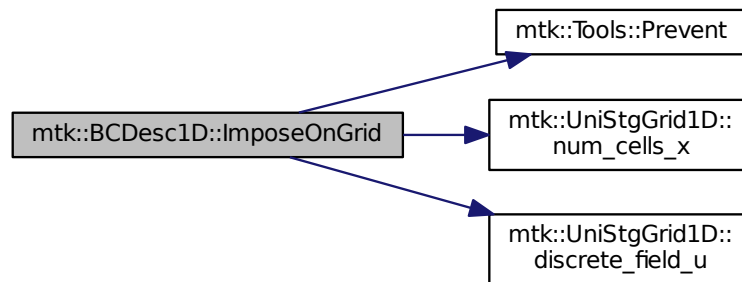
#### 16.1.2 Member Function Documentation

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

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

Definition at line 30 of file [mtk\\_bc\\_desc\\_1d.cc](#).

Here is the call graph for this function:

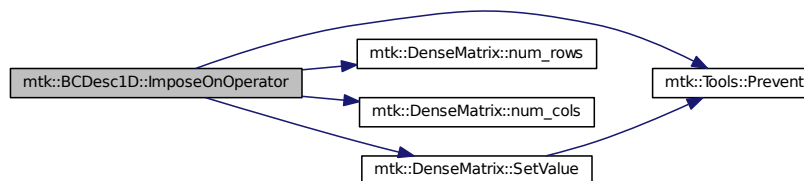


16.1.2.2 `void mtk::BCDesc1D::ImposeOnOperator ( mtk::DenseMatrix & matrix, const std::vector< Real > & west, const std::vector< Real > & east )`  
`[static]`

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

Definition at line 5 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](#)

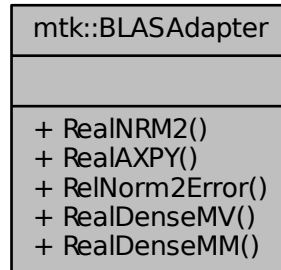


## 16.2 mtk::BLASAdapter Class Reference

Adapter class for the BLAS API.

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

Collaboration diagram for mtk::BLASAdapter:



### Static Public Member Functions

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

### 16.2.1 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](#).

## 16.2.2 Member Function Documentation

16.2.2.1 `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

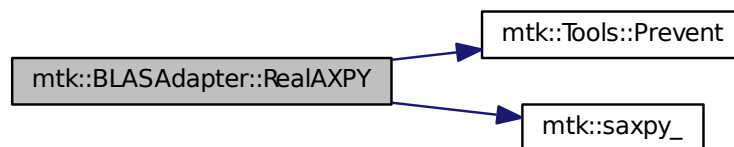
in	<i>alpha</i>	Scalar of the first array.
in	<i>xx</i>	First array.
in	<i>yy</i>	Second array.
in	<i>in_length</i>	Lengths of the given arrays.

Returns

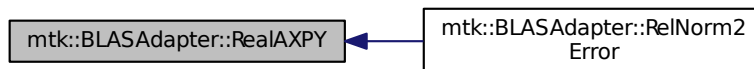
Norm-2 of the given array.

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:



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

Performs:

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

## Parameters

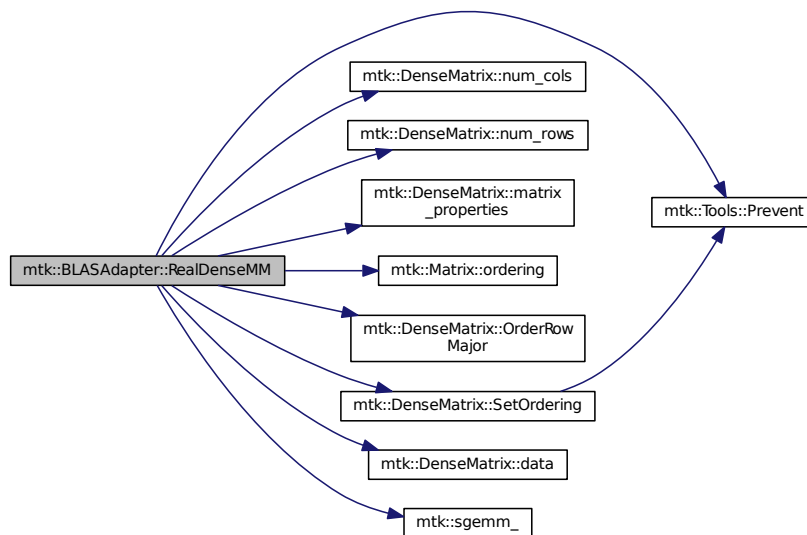
in	<i>aa</i>	First matrix.
in	<i>bb</i>	Second matrix.

## See Also

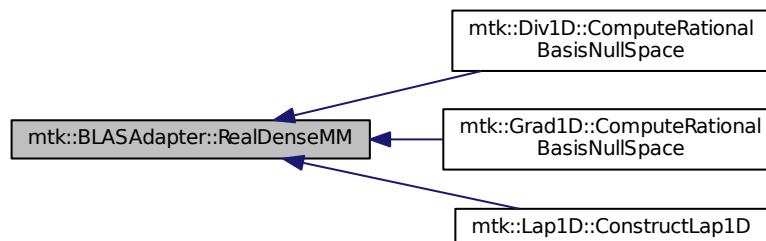
<http://ejspeiro.github.io/Netlib-and-CPP/>

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:



16.2.2.3 `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

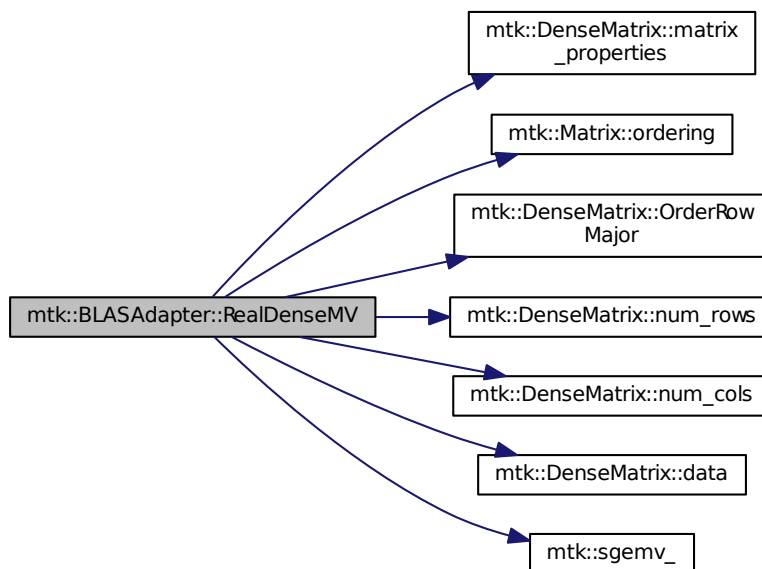
in	<i>alpha</i>	First scalar.
in	<i>aa</i>	Given matrix.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).

#### See Also

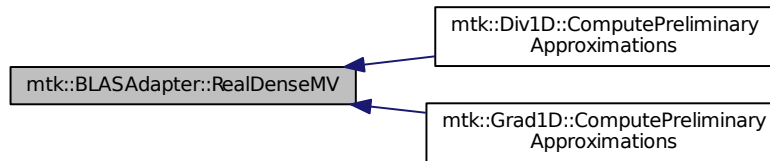
<http://ejspeiro.github.io/Netlib-and-CPP/>

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:



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

##### Parameters

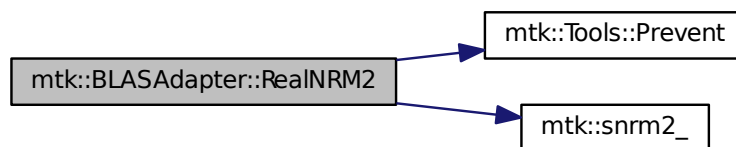
<code>in</code>	<code>in</code>	Input array.
<code>in</code>	<code>in_length</code>	Length of the array.

##### Returns

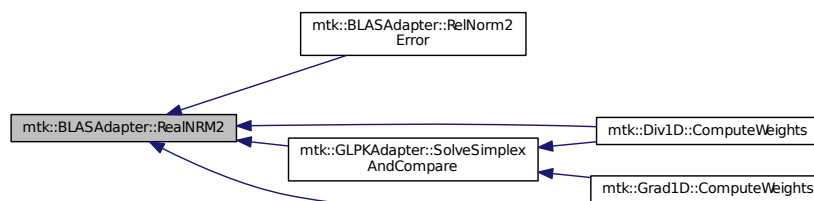
Norm-2 of the given array.

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:



16.2.2.5 `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

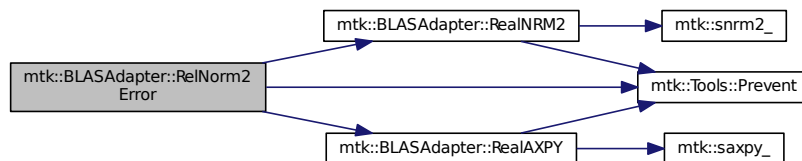
in	<i>known</i>	Array containing the computed solution.
in	<i>computed</i>	Array containing the known solution (ref. solution).

#### 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:

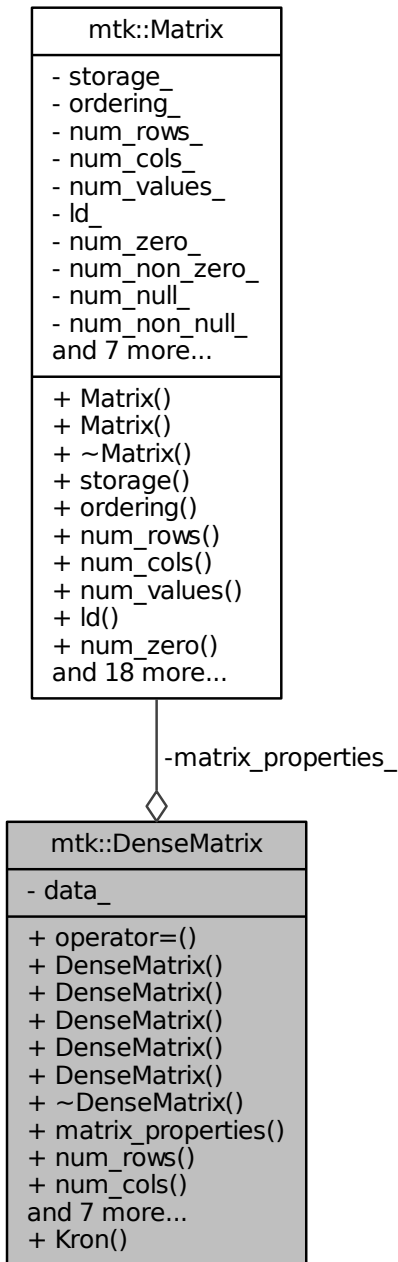
- [include/mtk\\_blas\\_adapter.h](#)
- [src/mtk\\_blas\\_adapter.cc](#)

## 16.3 mtk::DenseMatrix Class Reference

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

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

Collaboration diagram for mtk::DenseMatrix:



## Public Member Functions

- `DenseMatrix` & `operator=` (const `DenseMatrix` &in)



- Overloaded assignment operator.*
- [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.*

## 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).*

### 16.3.1 Detailed Description

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

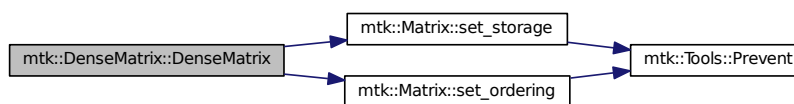
Definition at line 92 of file [mtk\\_dense\\_matrix.h](#).

### 16.3.2 Constructor & Destructor Documentation

#### 16.3.2.1 `mtk::DenseMatrix::DenseMatrix ( )`

Definition at line 138 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



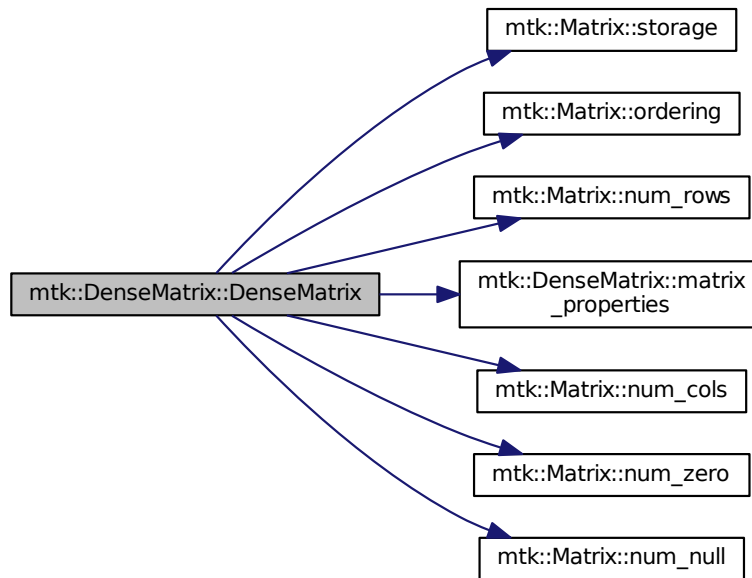
#### 16.3.2.2 `mtk::DenseMatrix::DenseMatrix ( const DenseMatrix &in )`

##### Parameters

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

Definition at line 144 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



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

##### Parameters

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

##### Exceptions

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

Definition at line 177 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



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

Used in the construction of the mimetic operators.

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

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

##### Parameters

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

##### Exceptions

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

Definition at line 199 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



#### 16.3.2.5 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

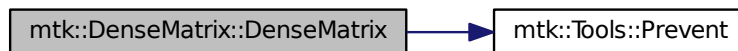
in	<i>gen</i>	Given generator vector.
in	<i>gen_length</i>	Length generator vector.
in	<i>pro_length</i>	Length the progression.
in	<i>transpose</i>	Should the transpose be created instead?

## Exceptions

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

Definition at line 237 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



## 16.3.2.6 mtk::DenseMatrix::~~DenseMatrix ( )

Definition at line 285 of file [mtk\\_dense\\_matrix.cc](#).

## 16.3.3 Member Function Documentation

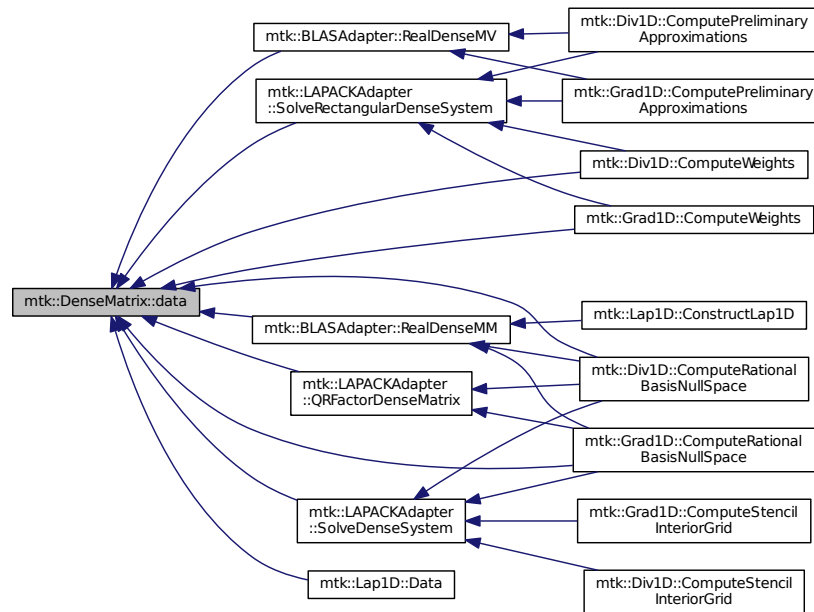
## 16.3.3.1 mtk::Real \* mtk::DenseMatrix::data ( ) const

## Returns

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

Definition at line 316 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



### 16.3.3.2 `mtk::Real mtk::DenseMatrix::GetValue ( const int & row_coord, const int & col_coord ) const`

## Parameters

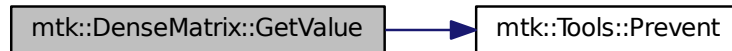
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.

**Returns**

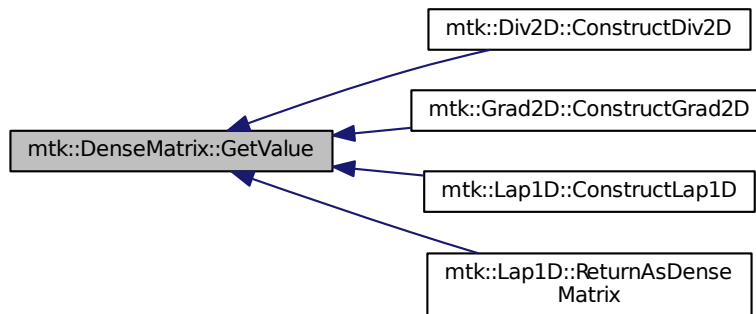
The required value at the specified coordinates.

Definition at line 321 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

**Parameters**

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

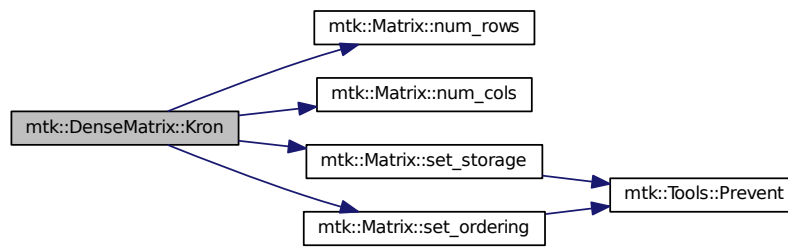
**Exceptions**

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

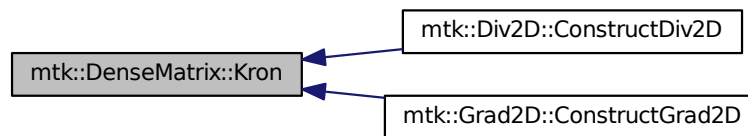
**Todo** Implement Kronecker product using the BLAS.

Definition at line 463 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



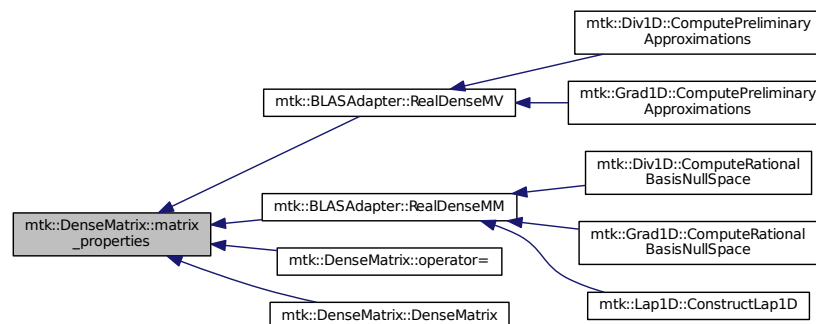
#### 16.3.3.4 mtk::Matrix mtk::DenseMatrix::matrix\_properties ( ) const

Returns

Pointer to a [Matrix](#).

Definition at line 291 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:





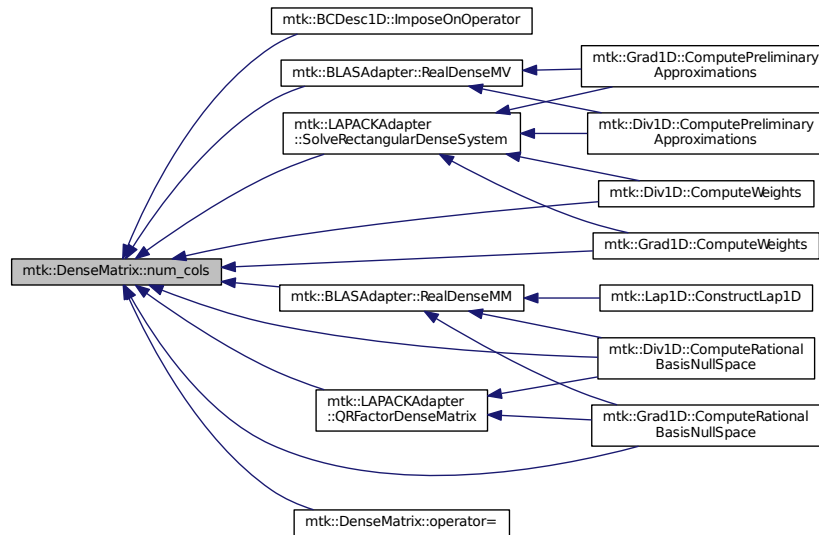
### 16.3.3.5 int mtk::DenseMatrix::num\_cols ( ) const

#### Returns

Number of columns of the matrix.

Definition at line 311 of file [mtk\\_dense\\_matrix.cc](#).

Here is the caller graph for this function:



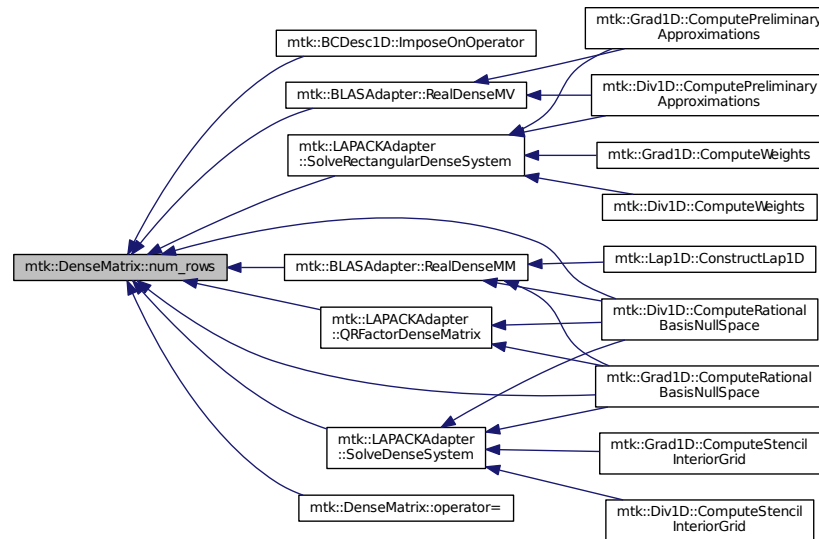
### 16.3.3.6 int mtk::DenseMatrix::num\_rows ( ) const

## Returns

Number of rows of the matrix.

Definition at line 306 of file [mtk\\_dense\\_matrix.cc](#).

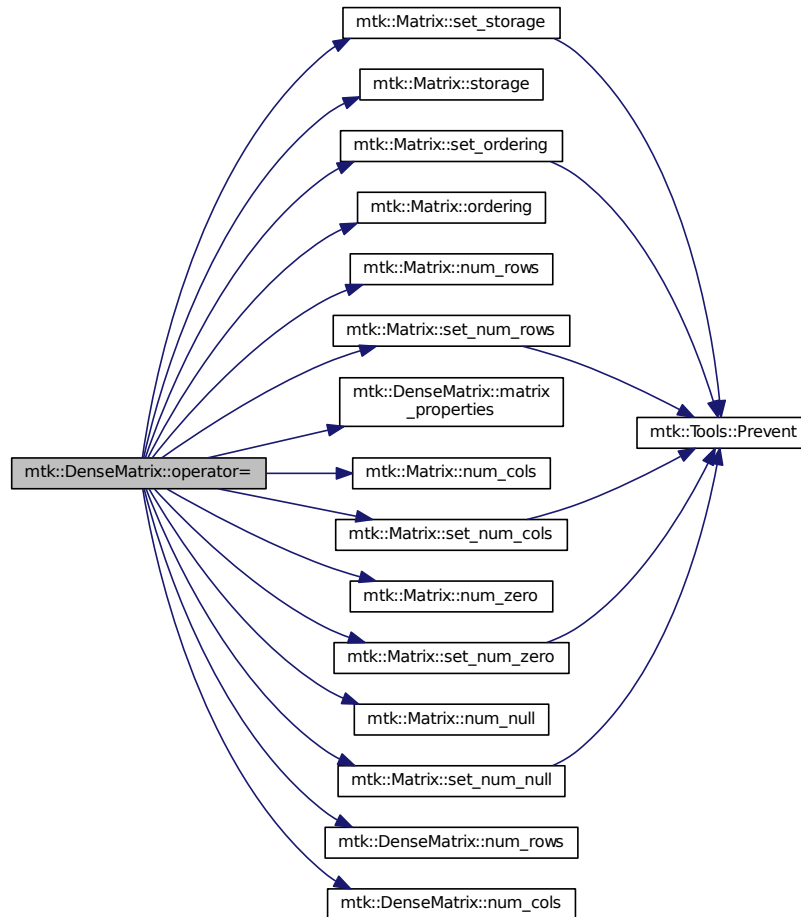
Here is the caller graph for this function:



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

Definition at line 97 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:

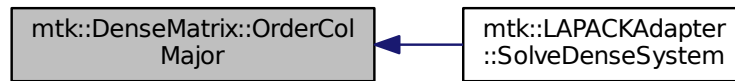


16.3.3.8 void mtk::DenseMatrix::OrderColMajor ( )

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

Definition at line 424 of file `mtk_dense_matrix.cc`.

Here is the caller graph for this function:

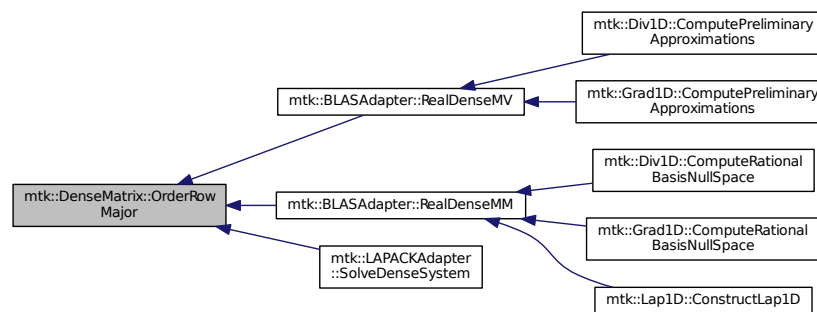


16.3.3.9 `void mtk::DenseMatrix::OrderRowMajor ( )`

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

Definition at line 383 of file `mtk_dense_matrix.cc`.

Here is the caller graph for this function:



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

#### Parameters

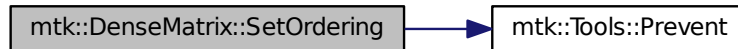
in	oo	Ordering.
----	----	-----------

**Returns**

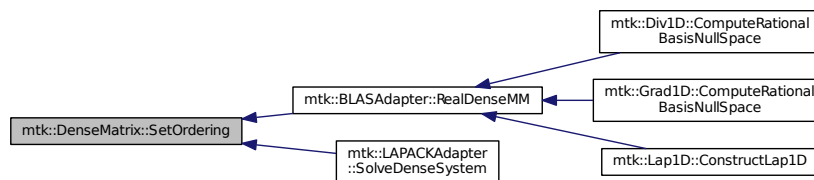
The required value at the specified coordinates.

Definition at line 296 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



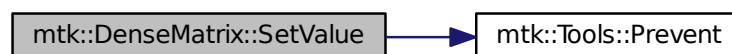
16.3.3.11 void `mtk::DenseMatrix::SetValue` ( const int & *row\_coord*, const int & *col\_coord*, const Real & *val* )

**Parameters**

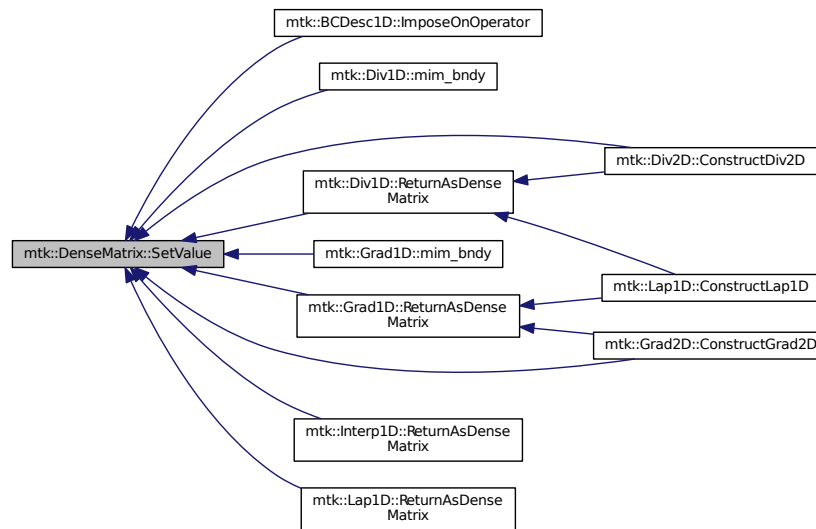
in	<i>row_coord</i>	Row coordinate.
in	<i>col_coord</i>	Column coordinate.
in	<i>val</i>	Row Actual value to be inserted.

Definition at line 333 of file [mtk\\_dense\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:

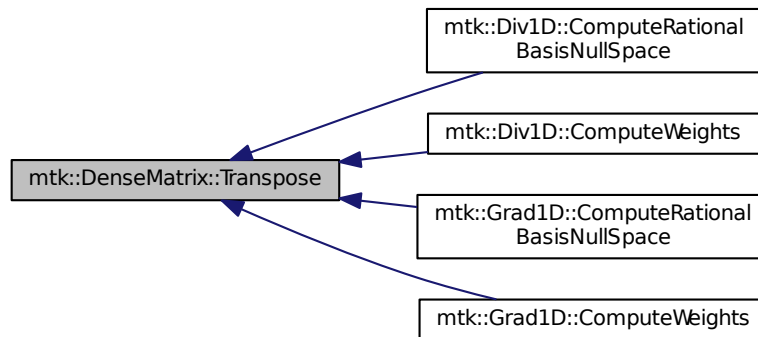


#### 16.3.3.12 void mtk::DenseMatrix::Transpose ( )

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

Definition at line 346 of file `mtk_dense_matrix.cc`.

Here is the caller graph for this function:



### 16.3.4 Friends And Related Function Documentation

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

Definition at line 75 of file [mtk\\_dense\\_matrix.cc](#).

## 16.3.5 Member Data Documentation

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

Definition at line 270 of file [mtk\\_dense\\_matrix.h](#).

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

Definition at line 268 of file [mtk\\_dense\\_matrix.h](#).

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

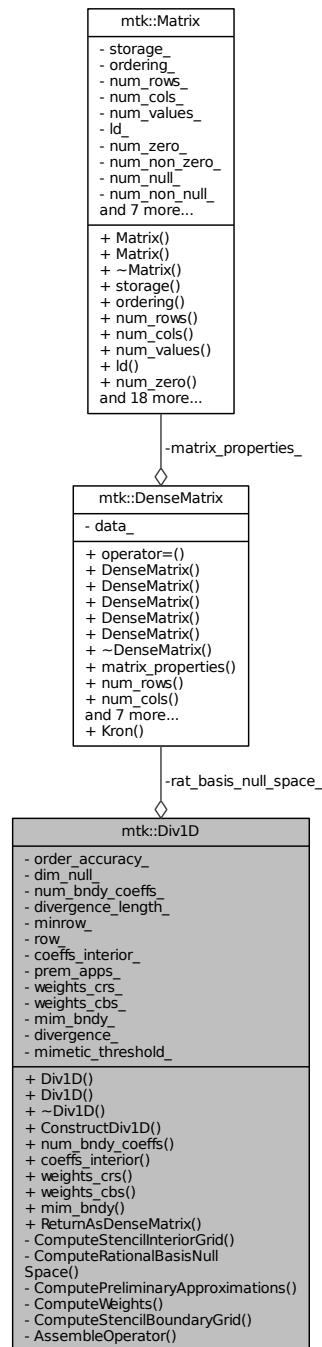
- [include/mtk\\_dense\\_matrix.h](#)
- [src/mtk\\_dense\\_matrix.cc](#)

## 16.4 mtk::Div1D Class Reference

Implements a 1D mimetic divergence operator.

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

Collaboration diagram for mtk::Div1D:



## Public Member Functions

- [Div1D \(\)](#)



- *Default constructor.*
- [Div1D](#) (const [Div1D](#) &div)
- *Copy constructor.*
- [~Div1D](#) ()
- *Destructor.*
- bool [ConstructDiv1D](#) (int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_threshold=[kDefaultMimeticThreshold](#))
- *Factory method implementing the CBS Algorithm to build operator.*
- int [num\\_bndy\\_coeffs](#) () const
- *Returns how many coefficients are approximating at the boundary.*
- [Real](#) \* [coeffs\\_interior](#) () const
- *Returns coefficients for the interior of the grid.*
- [Real](#) \* [weights\\_crs](#) (void) const
- *Return collection of weights as computed by the CRSA.*
- [Real](#) \* [weights\\_cbs](#) (void) const
- *Return collection of weights as computed by the CBSA.*
- [DenseMatrix](#) [mim\\_bndy](#) () const
- *Return collection of mimetic approximations at the boundary.*
- [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid)
- *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\\_coeffs\\_](#)
- *Req. coeffs. per bndy pt. uni. order accuracy.*
- int [divergence\\_length\\_](#)
- *Length of the output array.*
- int [minrow\\_](#)

- *Row from the optimizer with the minimum rel. nor.*
- `int row_`  
*Row currently processed by the optimizer.*
- `DenseMatrix rat_basis_null_space_`  
*Rational b. null-space w. bndy.*
- `Real * coeffs_interior_`  
*Interior stencil.*
- `Real * prem_apps_`  
*2D array of boundary preliminary approximations.*
- `Real * weights_crs_`  
*Array containing weights from CRSA.*
- `Real * weights_cbs_`  
*Array containing weights from CBSA.*
- `Real * mim_bndy_`  
*Array containing mimetic boundary approximations.*
- `Real * divergence_`  
*Output array containing the operator and weights.*
- `Real mimetic_threshold_`  
*< Mimetic threshold.*

## Friends

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

### 16.4.1 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`.

### 16.4.2 Constructor & Destructor Documentation

#### 16.4.2.1 `mtk::Div1D::Div1D ( )`

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

#### 16.4.2.2 `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`.

#### 16.4.2.3 `mtk::Div1D::~~Div1D ( )`

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

### 16.4.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

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

Definition at line 1333 of file [mtk\\_div\\_1d.cc](#).

#### 16.4.3.2 `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](#).

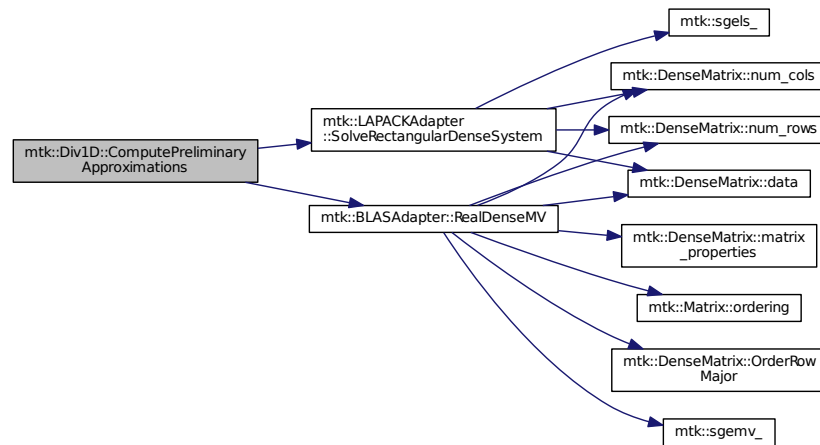
#### 16.4.3.3 `bool mtk::Div1D::ComputePreliminaryApproximations ( void ) [private]`

Compute the set of preliminary approximations on the boundary neighborhood.

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



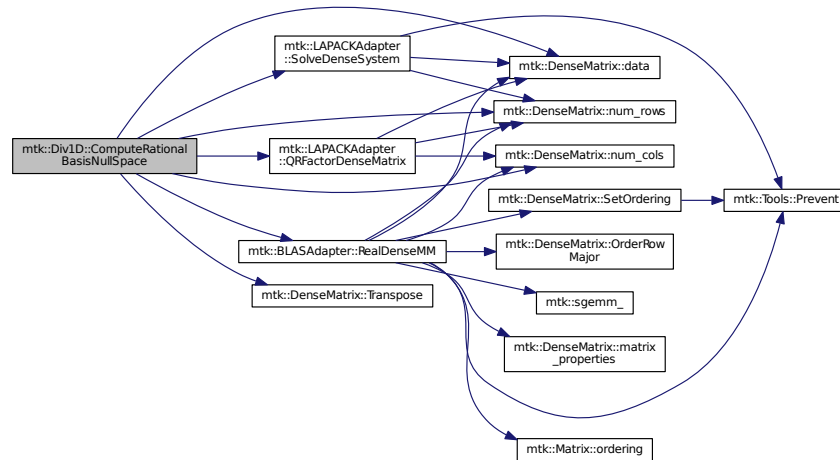
#### 16.4.3.4 `bool mtk::Div1D::ComputeRationalBasisNullSpace ( void ) [private]`

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

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

Definition at line 512 of file `mtk_div_1d.cc`.

Here is the call graph for this function:



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

Compute mimetic stencil approximating at boundary.

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

Definition at line 1234 of file [mtk\\_div\\_1d.cc](#).

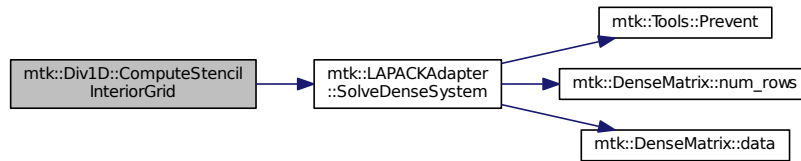
#### 16.4.3.6 bool mtk::Div1D::ComputeStencilInteriorGrid ( void ) [private]

Compute the stencil approximating the interior of the staggered grid.

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

Definition at line 413 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



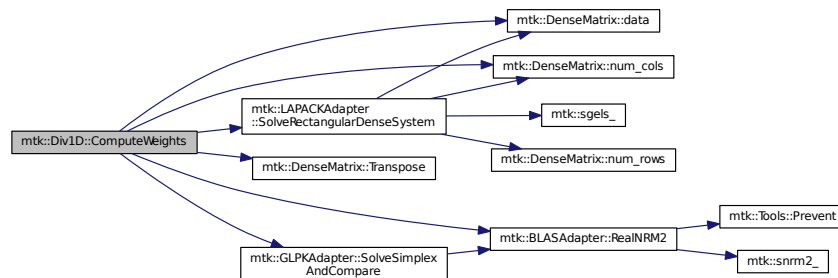
#### 16.4.3.7 bool mtk::Div1D::ComputeWeights ( void ) [private]

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

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

Definition at line 908 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



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

#### Returns

Success of the construction.

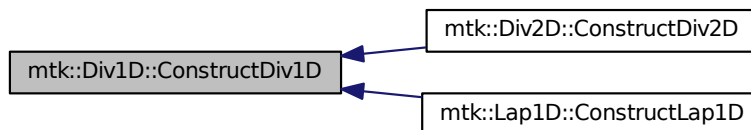
1. Compute stencil for the interior cells.
2. Compute a rational basis for the null-space for the first matrix.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 176 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



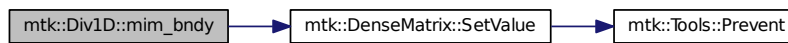
16.4.3.9 `mtk::DenseMatrix mtk::Div1D::mim_bndy ( ) const`

**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:



#### 16.4.3.10 `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](#).

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

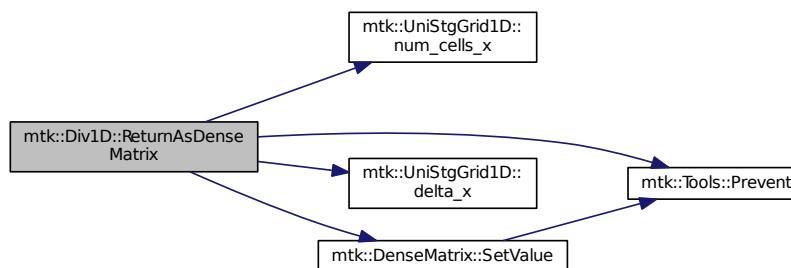
**Returns**

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.

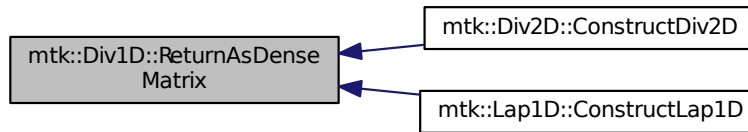
Definition at line 351 of file [mtk\\_div\\_1d.cc](#).

Here is the call graph for this function:





Here is the caller graph for this function:



16.4.3.12 `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](#).

16.4.3.13 `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](#).

## 16.4.4 Friends And Related Function Documentation

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

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

Definition at line 79 of file [mtk\\_div\\_1d.cc](#).

## 16.4.5 Member Data Documentation

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

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

16.4.5.2 `int mtk::Div1D::dim_null_ [private]`

Definition at line 194 of file [mtk\\_div\\_1d.h](#).

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

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

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

Definition at line 196 of file [mtk\\_div\\_1d.h](#).

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

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

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

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

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

Definition at line 197 of file [mtk\\_div\\_1d.h](#).

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

Definition at line 195 of file [mtk\\_div\\_1d.h](#).

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

Definition at line 193 of file [mtk\\_div\\_1d.h](#).

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

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

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

Definition at line 200 of file [mtk\\_div\\_1d.h](#).

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

Definition at line 198 of file [mtk\\_div\\_1d.h](#).

#### 16.4.5.13 Real\* mtk::Div1D::weights\_cbs\_ [private]

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

#### 16.4.5.14 Real\* mtk::Div1D::weights\_crs\_ [private]

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

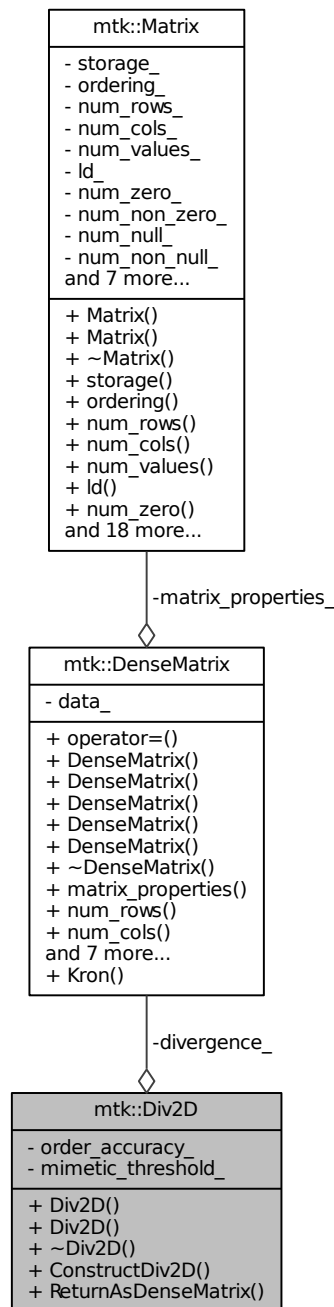
The documentation for this class was generated from the following files:

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

## 16.5 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.*

- [DenseMatrix 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.*

## 16.5.1 Detailed Description

Definition at line 66 of file [mtk\\_div\\_2d.h](#).

## 16.5.2 Constructor & Destructor Documentation

### 16.5.2.1 mtk::Div2D::Div2D ( )

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

### 16.5.2.2 mtk::Div2D::Div2D ( const Div2D & div )

#### Parameters

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

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

### 16.5.2.3 mtk::Div2D::~~Div2D ( )

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

## 16.5.3 Member Function Documentation

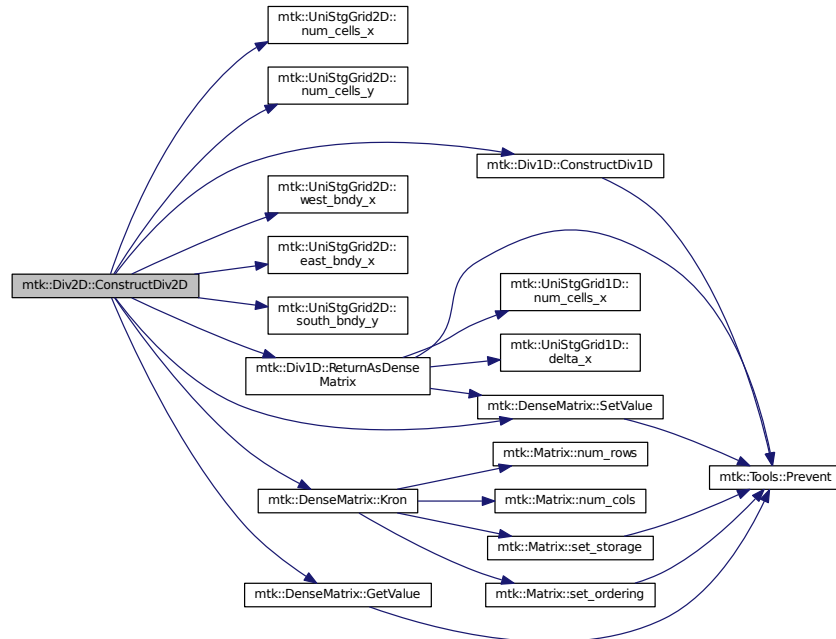
### 16.5.3.1 mtk::DenseMatrix mtk::Div2D::ConstructDiv2D ( const UniStgGrid2D & grid, int order\_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic\_threshold = kDefaultMimeticThreshold )

**Returns**

Success of the construction.

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

Here is the call graph for this function:



### 16.5.3.2 `mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix ( )`

**Returns**

The operator as a dense matrix.

Definition at line 144 of file [mtk\\_div\\_2d.cc](#).

## 16.5.4 Member Data Documentation

### 16.5.4.1 `DenseMatrix mtk::Div2D::divergence_ [private]`

Definition at line 98 of file [mtk\\_div\\_2d.h](#).

### 16.5.4.2 `Real mtk::Div2D::mimetic_threshold_ [private]`

Definition at line 100 of file [mtk\\_div\\_2d.h](#).

16.5.4.3 `int mtk::Div2D::order_accuracy_ [private]`

Definition at line 99 of file [mtk\\_div\\_2d.h](#).

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

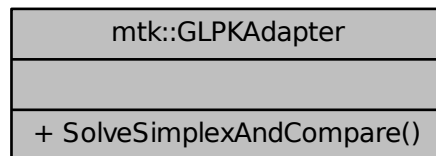
- [include/mtk\\_div\\_2d.h](#)
- [src/mtk\\_div\\_2d.cc](#)

## 16.6 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.*

### 16.6.1 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](#).

## 16.6.2 Member Function Documentation

**16.6.2.1** `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.

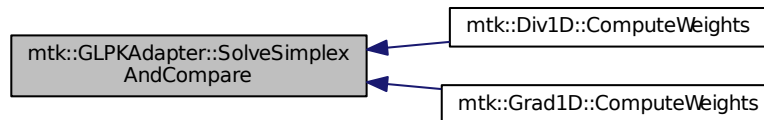
Definition at line 76 of file [mtk\\_glpk\\_adapter.cc](#).



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:

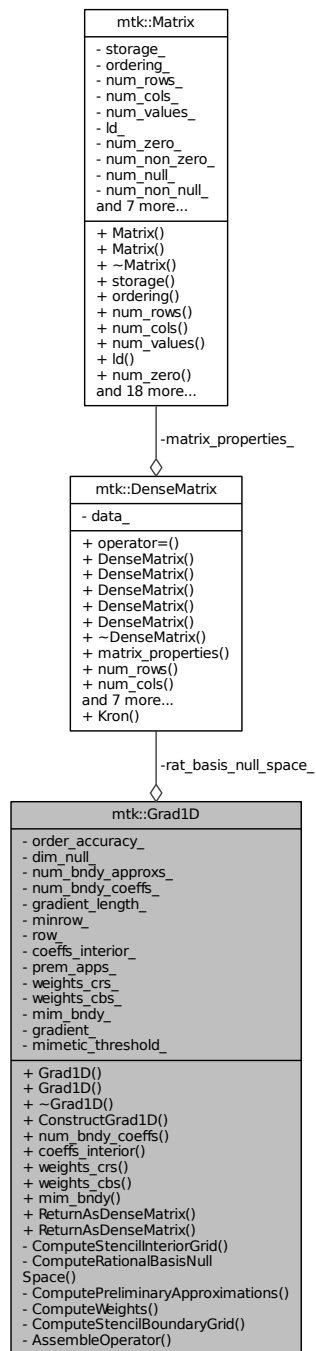
- [include/mtk\\_glpk\\_adapter.h](#)
- [src/mtk\\_glpk\\_adapter.cc](#)

## 16.7 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.*

### Private Member Functions

- bool [ComputeStencillInteriorGrid](#) (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.*

### 16.7.1 Detailed Description

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

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

### 16.7.2 Constructor & Destructor Documentation

#### 16.7.2.1 [mtk::Grad1D::Grad1D \( \)](#)

Definition at line 129 of file [mtk\\_grad\\_1d.cc](#).

#### 16.7.2.2 [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](#).

## 16.7.2.3 mtk::Grad1D::~~Grad1D ( )

Definition at line 161 of file [mtk\\_grad\\_1d.cc](#).

## 16.7.3 Member Function Documentation

## 16.7.3.1 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  $\text{dim\_null} + 1$  entries will contain the collections of approximating coefficients for the west boundary of the grid.

Definition at line 1437 of file [mtk\\_grad\\_1d.cc](#).

## 16.7.3.2 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](#).

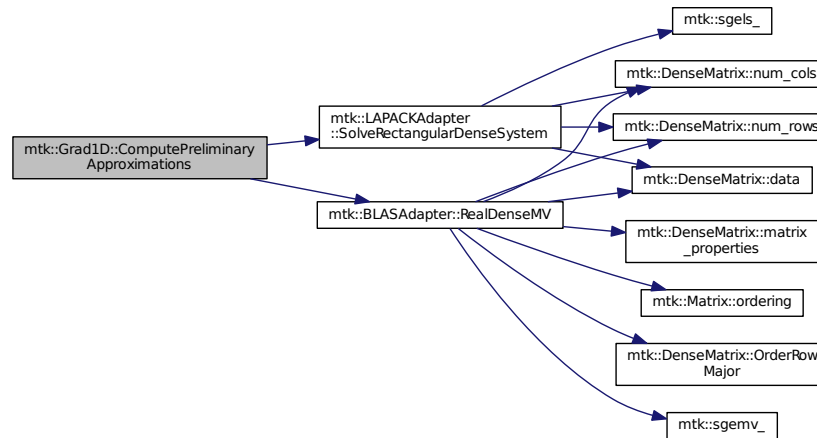
## 16.7.3.3 bool mtk::Grad1D::ComputePreliminaryApproximations ( void ) [private]

Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.
2. Compute the  $\text{dim\_null}$  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  $\text{TT} * \text{rr} = \text{ob}$  yields the columns  $\text{rr}$  of the  $\text{kk}$  matrix.
6. Scale the  $\text{kk}$  matrix to make it a rational basis for null-space.
7. Extract the last  $\text{dim\_null}$  values of the pre-scaled  $\text{ob}$ .
8. Once we possess the bottom elements, we proceed with the scaling.

Definition at line 771 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



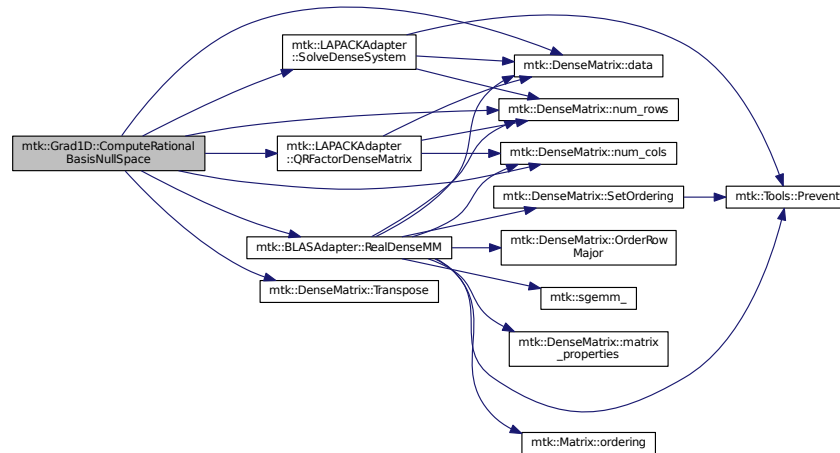
#### 16.7.3.4 `bool mtk::Grad1D::ComputeRationalBasisNullSpace ( void ) [private]`

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 588 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



#### 16.7.3.5 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 1331 of file [mtk\\_grad\\_1d.cc](#).

#### 16.7.3.6 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 492 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



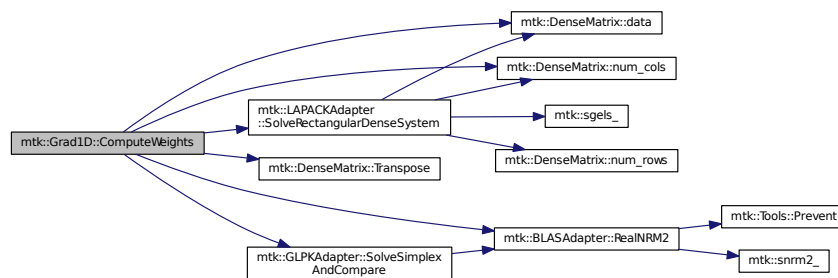
### 16.7.3.7 bool mtk::Grad1D::ComputeWeights ( void ) [private]

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

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

Definition at line 991 of file [mtk\\_grad\\_1d.cc](#).

Here is the call graph for this function:



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



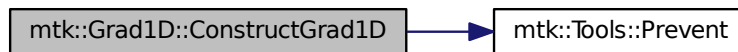
**Returns**

Success of the solution.

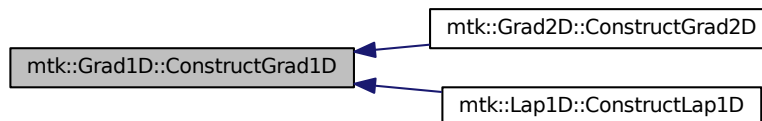
1. Compute stencil for the interior cells.
2. Compute a rational null-space from the first matrix transposed.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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



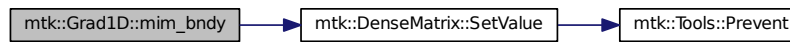
### 16.7.3.9 mtk::DenseMatrix mtk::Grad1D::mim\_bndy ( ) const

**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:



### 16.7.3.10 `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](#).

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

**Returns**

The operator as a dense matrix.

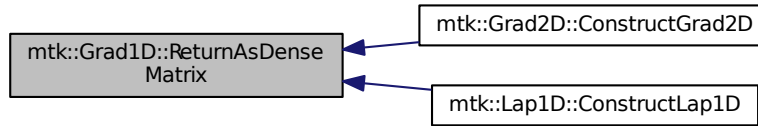
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

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:



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

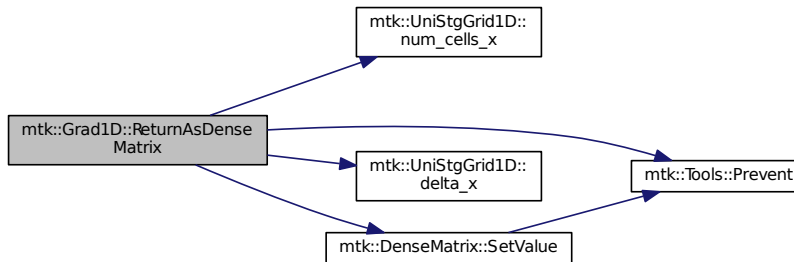
##### Returns

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:



#### 16.7.3.13 `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](#).

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

#### Returns

Success of the solution.

Definition at line 335 of file [mtk\\_grad\\_1d.cc](#).

### 16.7.4 Friends And Related Function Documentation

16.7.4.1 `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](#).

### 16.7.5 Member Data Documentation

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

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

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

Definition at line 201 of file [mtk\\_grad\\_1d.h](#).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Definition at line 200 of file [mtk\\_grad\\_1d.h](#).

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

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

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

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

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

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

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

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

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

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

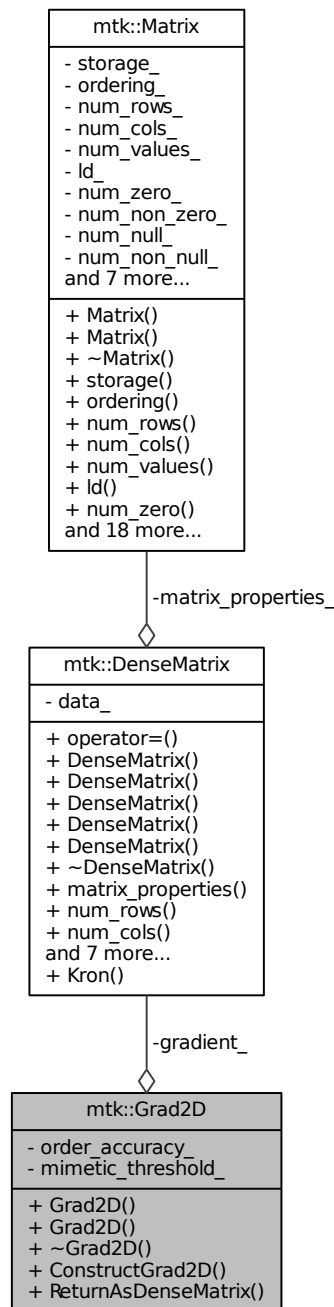
The documentation for this class was generated from the following files:

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

## 16.8 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.*

- [DenseMatrix 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.*

## 16.8.1 Detailed Description

Definition at line 66 of file [mtk\\_grad\\_2d.h](#).

## 16.8.2 Constructor & Destructor Documentation

### 16.8.2.1 mtk::Grad2D::Grad2D ( )

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

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

#### Parameters

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

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

### 16.8.2.3 mtk::Grad2D::~~Grad2D ( )

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

## 16.8.3 Member Function Documentation

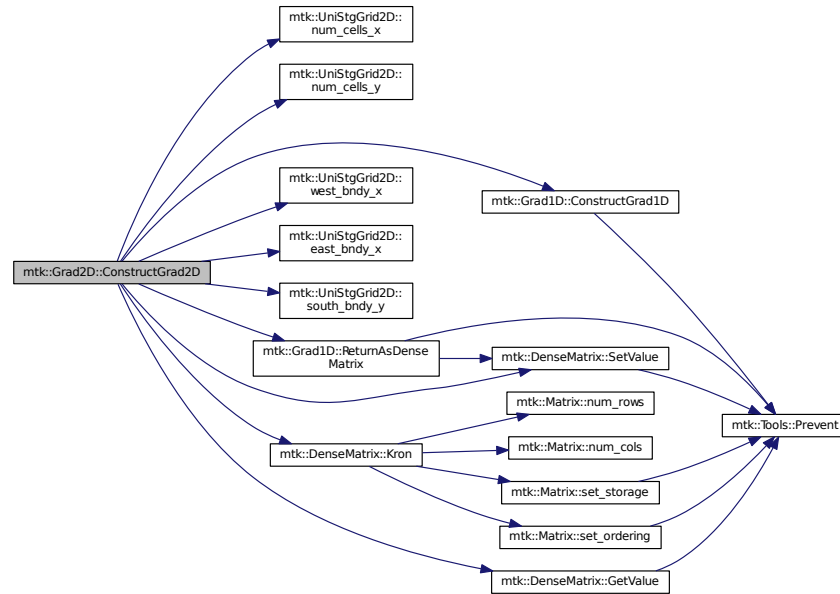
### 16.8.3.1 mtk::DenseMatrix mtk::Grad2D::ConstructGrad2D ( const UniStgGrid2D & grid, int order\_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic\_threshold = kDefaultMimeticThreshold )

**Returns**

Success of the construction.

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

Here is the call graph for this function:

**16.8.3.2 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix ( )****Returns**

The operator as a dense matrix.

Definition at line 142 of file [mtk\\_grad\\_2d.cc](#).

**16.8.4 Member Data Documentation****16.8.4.1 DenseMatrix mtk::Grad2D::gradient\_ [private]**

Definition at line 98 of file [mtk\\_grad\\_2d.h](#).

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

Definition at line 100 of file [mtk\\_grad\\_2d.h](#).

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

Definition at line 99 of file [mtk\\_grad\\_2d.h](#).



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

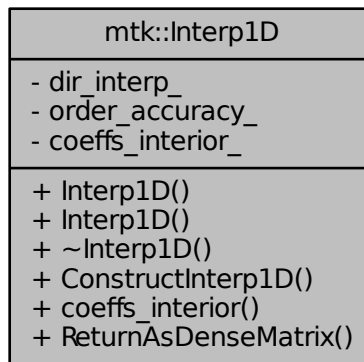
- [include/mtk\\_grad\\_2d.h](#)
- [src/mtk\\_grad\\_2d.cc](#)

## 16.9 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

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

Collaboration diagram for mtk::Interp1D:



### Public Member Functions

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

### 16.9.1 Detailed Description

This class implements a 1D interpolation operator.

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

### 16.9.2 Constructor & Destructor Documentation

#### 16.9.2.1 `mtk::Interp1D::Interp1D ( )`

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

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

##### Parameters

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

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

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

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

### 16.9.3 Member Function Documentation

#### 16.9.3.1 `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](#).

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

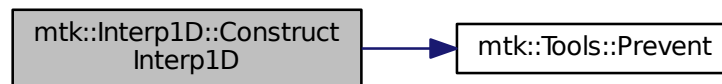
#### Returns

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:



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

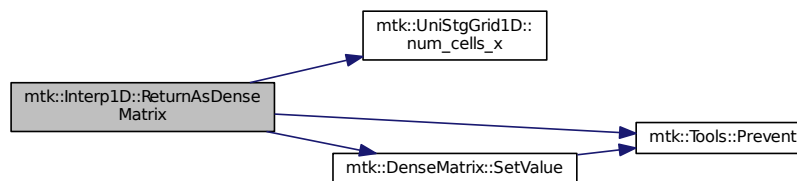
#### Returns

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:



## 16.9.4 Friends And Related Function Documentation

16.9.4.1 `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](#).

## 16.9.5 Member Data Documentation

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

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

16.9.5.2 `DirInterp mtk::Interp1D::dir_interp_` [*private*]

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

16.9.5.3 `int mtk::Interp1D::order_accuracy_` [*private*]

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

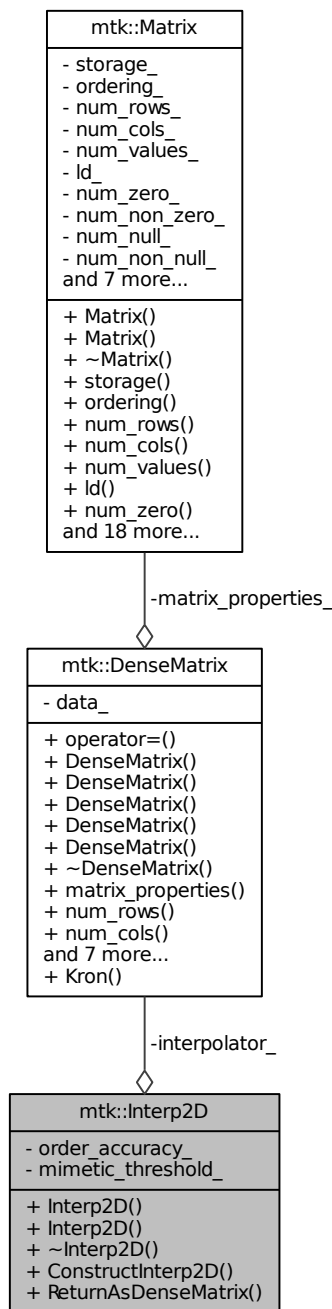
The documentation for this class was generated from the following files:

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

## 16.10 mtk::Interp2D Class Reference

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

Collaboration diagram for mtk::Interp2D:



## Public Member Functions

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

- [~Interp2D](#) ()
- [DenseMatrix ConstructInterp2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_threshold=[kDefaultMimeticThreshold](#))
- [DenseMatrix ReturnAsDenseMatrix](#) ()

### Private Attributes

- [DenseMatrix interpolator\\_](#)
- int [order\\_accuracy\\_](#)
- [Real mimetic\\_threshold\\_](#)

### 16.10.1 Detailed Description

Definition at line 67 of file [mtk\\_interp\\_2d.h](#).

### 16.10.2 Constructor & Destructor Documentation

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

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

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

### 16.10.3 Member Function Documentation

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

16.10.3.2 [DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix](#) ( )

### 16.10.4 Member Data Documentation

16.10.4.1 [DenseMatrix mtk::Interp2D::interpolator\\_](#) [private]

Definition at line 78 of file [mtk\\_interp\\_2d.h](#).

16.10.4.2 [Real mtk::Interp2D::mimetic\\_threshold\\_](#) [private]

Definition at line 80 of file [mtk\\_interp\\_2d.h](#).

16.10.4.3 [int mtk::Interp2D::order\\_accuracy\\_](#) [private]

Definition at line 79 of file [mtk\\_interp\\_2d.h](#).

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

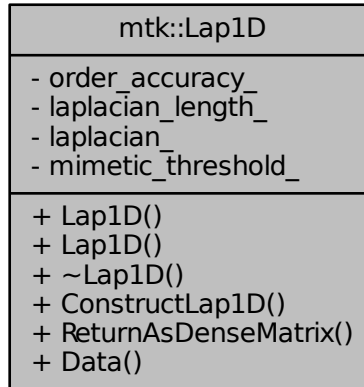
- [include/mtk\\_interp\\_2d.h](#)

## 16.11 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=kDefaultMimetic-Threshold)  
*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.*

### 16.11.1 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](#).

### 16.11.2 Constructor & Destructor Documentation

#### 16.11.2.1 `mtk::Lap1D::Lap1D ( )`

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

#### 16.11.2.2 `mtk::Lap1D::Lap1D ( const Lap1D &lap )`

##### Parameters

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

#### 16.11.2.3 `mtk::Lap1D::~~Lap1D ( )`

Definition at line 113 of file [mtk\\_lap\\_1d.cc](#).

### 16.11.3 Member Function Documentation

#### 16.11.3.1 `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.



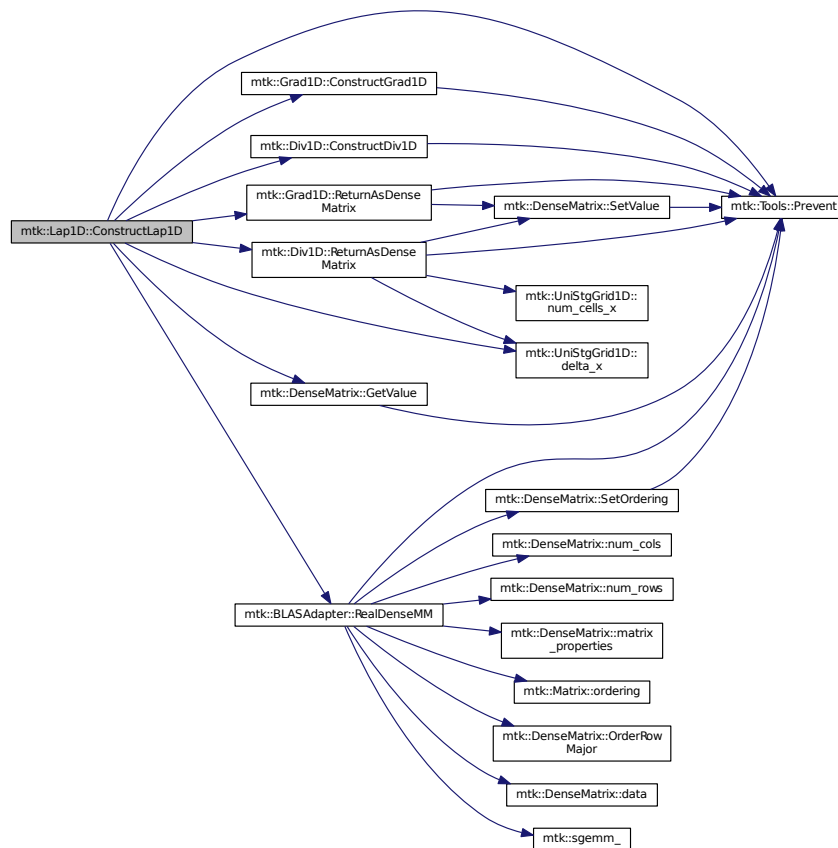
**Warning**

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:



### 16.11.3.2 `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:



### 16.11.3.3 `mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix ( const UniStgGrid1D & grid )`

**Returns**

The operator as a dense matrix.

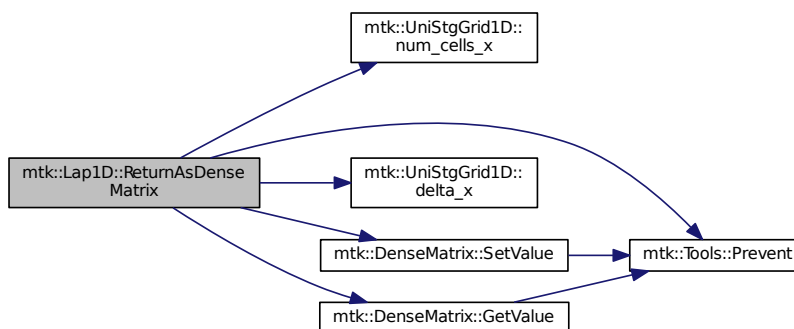
1. Extract mimetic coefficients from the west boundary.
2. Extract interior coefficients.
3. Extract mimetic coefficients from the west boundary to go east.

**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:



### 16.11.4 Friends And Related Function Documentation

16.11.4.1 `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](#).

### 16.11.5 Member Data Documentation

16.11.5.1 `Real* mtk::Lap1D::laplacian_` [*private*]

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

16.11.5.2 `int mtk::Lap1D::laplacian_length_` [*private*]

Definition at line 118 of file [mtk\\_lap\\_1d.h](#).

16.11.5.3 `Real mtk::Lap1D::mimetic_threshold_` [*private*]

Definition at line 122 of file [mtk\\_lap\\_1d.h](#).

16.11.5.4 `int mtk::Lap1D::order_accuracy_` [*private*]

Definition at line 117 of file [mtk\\_lap\\_1d.h](#).

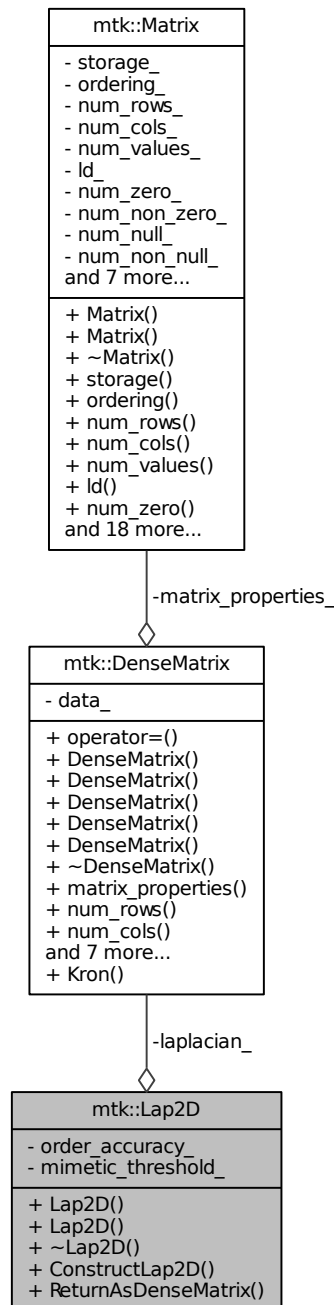
The documentation for this class was generated from the following files:

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

## 16.12 mtk::Lap2D Class Reference

```
#include <mtk_lap_2d.h>
```

Collaboration diagram for mtk::Lap2D:



## Public Member Functions

- [Lap2D](#) ()
- [Lap2D](#) (const [Lap2D](#) &lap)

- [~Lap2D](#) ()
- [DenseMatrix ConstructLap2D](#) (const [UniStgGrid2D](#) &grid, int order\_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic\_threshold=[kDefaultMimeticThreshold](#))
- [DenseMatrix ReturnAsDenseMatrix](#) ()

### Private Attributes

- [DenseMatrix laplacian\\_](#)
- int [order\\_accuracy\\_](#)
- [Real](#) [mimetic\\_threshold\\_](#)

### 16.12.1 Detailed Description

Definition at line 66 of file [mtk\\_lap\\_2d.h](#).

### 16.12.2 Constructor & Destructor Documentation

#### 16.12.2.1 mtk::Lap2D::Lap2D ( )

Definition at line 66 of file [mtk\\_lap\\_2d.cc](#).

#### 16.12.2.2 mtk::Lap2D::Lap2D ( const Lap2D & lap )

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

#### 16.12.2.3 mtk::Lap2D::~~Lap2D ( )

Definition at line 74 of file [mtk\\_lap\\_2d.cc](#).

### 16.12.3 Member Function Documentation

#### 16.12.3.1 mtk::DenseMatrix mtk::Lap2D::ConstructLap2D ( const UniStgGrid2D & grid, int order\_accuracy = [kDefaultOrderAccuracy](#), mtk::Real *mimetic\_threshold* = [kDefaultMimeticThreshold](#) )

Definition at line 76 of file [mtk\\_lap\\_2d.cc](#).

#### 16.12.3.2 DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix ( )

### 16.12.4 Member Data Documentation

#### 16.12.4.1 DenseMatrix mtk::Lap2D::laplacian\_ [private]

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

#### 16.12.4.2 Real mtk::Lap2D::mimetic\_threshold\_ [private]

Definition at line 79 of file [mtk\\_lap\\_2d.h](#).

16.12.4.3 `int mtk::Lap2D::order_accuracy_` [private]

Definition at line 78 of file [mtk\\_lap\\_2d.h](#).

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

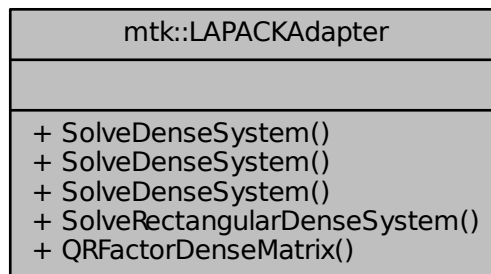
- [include/mtk\\_lap\\_2d.h](#)
- [src/mtk\\_lap\\_2d.cc](#)

## 16.13 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.*

### 16.13.1 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](#).

### 16.13.2 Member Function Documentation

16.13.2.1 `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

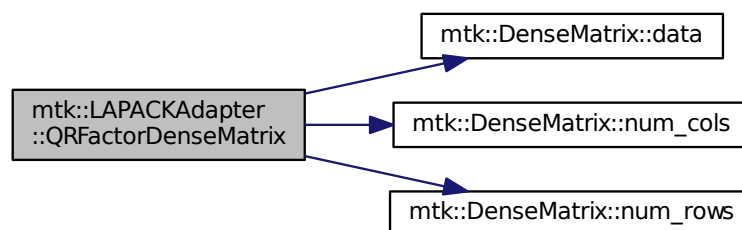
Matrix **Q**.

#### Exceptions

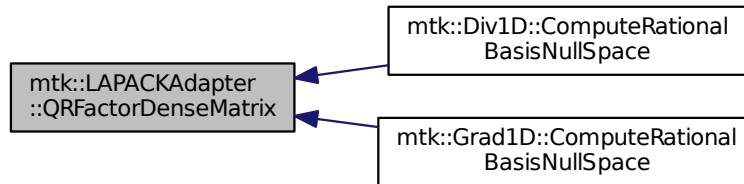
<i>std::bad_alloc</i>
-----------------------

Definition at line 553 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

#### Parameters

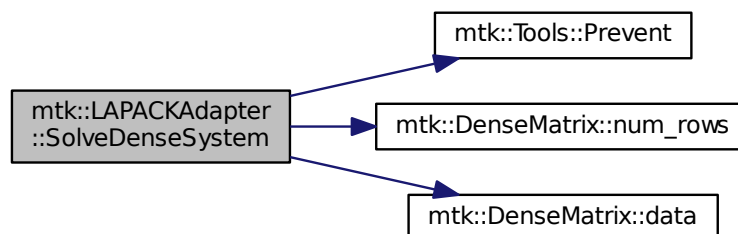
<code>in</code>	<code>matrix</code>	Input matrix.
<code>in</code>	<code>rhs</code>	Input right-hand sides vector.

#### Exceptions

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

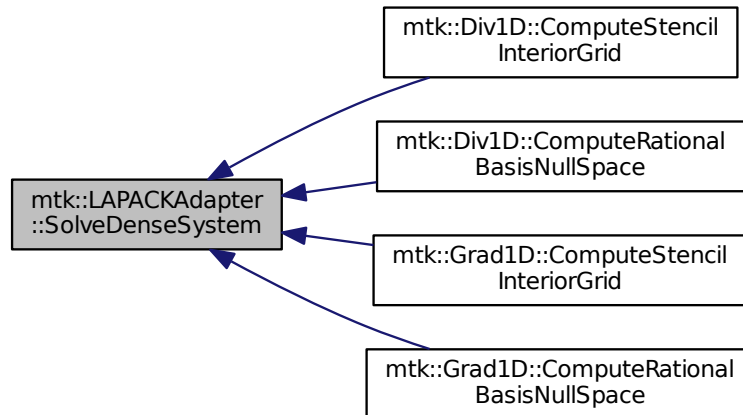
Definition at line 428 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:





Here is the caller graph for this function:



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

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

#### Parameters

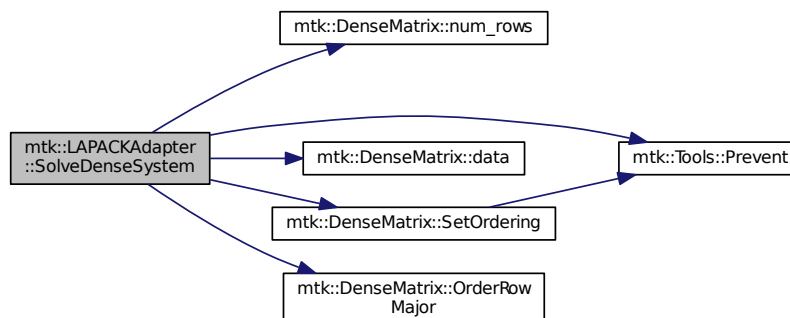
in	<i>matrix</i>	Input matrix.
in	<i>rr</i>	Input right-hand sides matrix.

#### Exceptions

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

Definition at line 463 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



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

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

#### Parameters

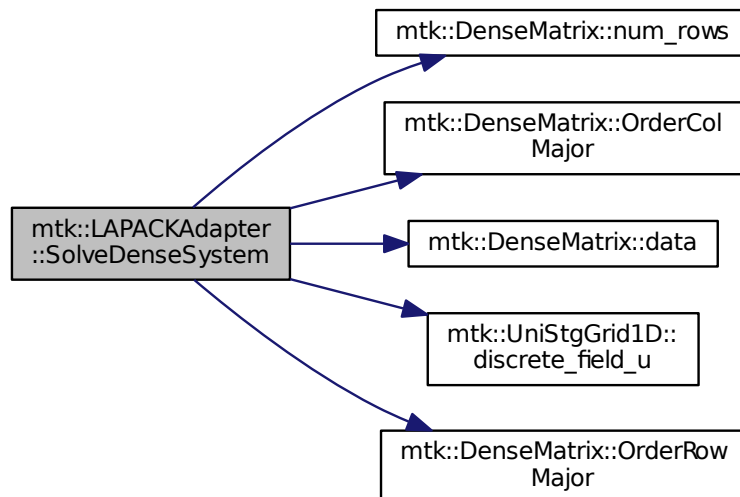
<code>in</code>	<code>matrix</code>	Input matrix.
<code>in</code>	<code>rr</code>	Input right-hand side from info on a grid.

#### Exceptions

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

Definition at line 515 of file [mtk\\_lapack\\_adapter.cc](#).

Here is the call graph for this function:



16.13.2.5 `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

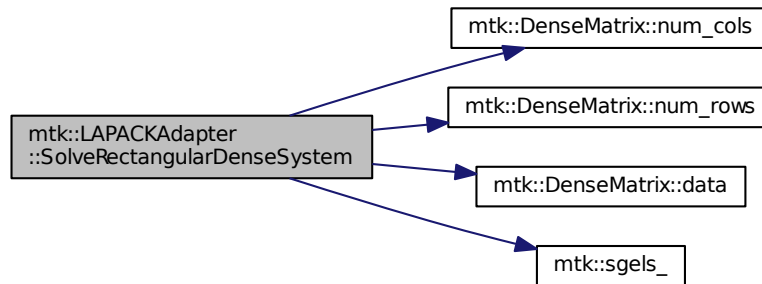
Success of the solution.

## Exceptions

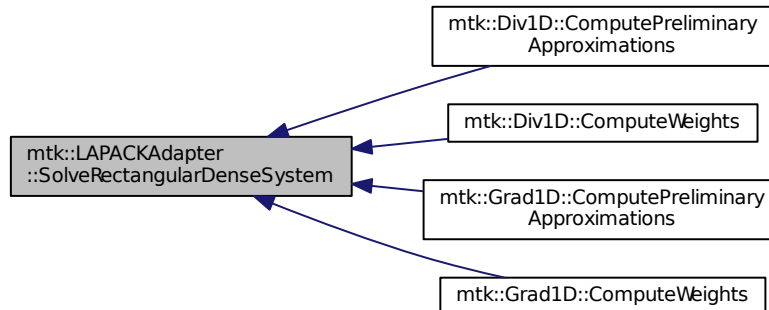
<code>std::bad_alloc</code>	
-----------------------------	--

Definition at line 754 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:

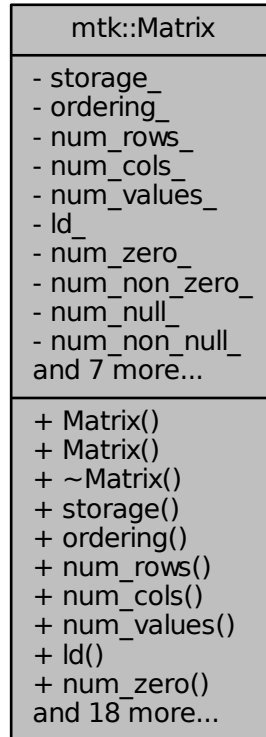
- [include/mtk\\_lapack\\_adapter.h](#)
- [src/mtk\\_lapack\\_adapter.cc](#)

## 16.14 mtk::Matrix Class Reference

Definition of the representation of a matrix in the MTK.

```
#include <mtk_matrix.h>
```

Collaboration diagram for mtk::Matrix:



## 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.*

### 16.14.1 Detailed Description

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

Definition at line 75 of file [mtk\\_matrix.h](#).

### 16.14.2 Constructor & Destructor Documentation

#### 16.14.2.1 `mtk::Matrix::Matrix ( )`

Definition at line 72 of file [mtk\\_matrix.cc](#).

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

## Parameters

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

Definition at line 91 of file [mtk\\_matrix.cc](#).

#### 16.14.2.3 `mtk::Matrix::~~Matrix ( )`

Definition at line 110 of file [mtk\\_matrix.cc](#).

### 16.14.3 Member Function Documentation

#### 16.14.3.1 `Real mtk::Matrix::abs_density ( ) const`

##### See Also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

##### Returns

Absolute density of the matrix.

#### 16.14.3.2 `mtk::Real mtk::Matrix::abs_sparsity ( ) const`

##### See Also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

##### Returns

Absolute sparsity of the matrix.

Definition at line 182 of file [mtk\\_matrix.cc](#).

#### 16.14.3.3 `int mtk::Matrix::bandwidth ( ) const`

##### Returns

Bandwidth of the matrix.

Definition at line 172 of file [mtk\\_matrix.cc](#).

#### 16.14.3.4 `void mtk::Matrix::IncreaseNumNull ( )`

**Todo** Review the definition of sparse matrices properties.

Definition at line 279 of file [mtk\\_matrix.cc](#).



16.14.3.5 void mtk::Matrix::IncreaseNumZero ( )

**Todo** Review the definition of sparse matrices properties.

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

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

Returns

Number of lower diagonals.

Definition at line 162 of file [mtk\\_matrix.cc](#).

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

Returns

Number of upper diagonals.

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

16.14.3.8 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 137 of file [mtk\\_matrix.cc](#).

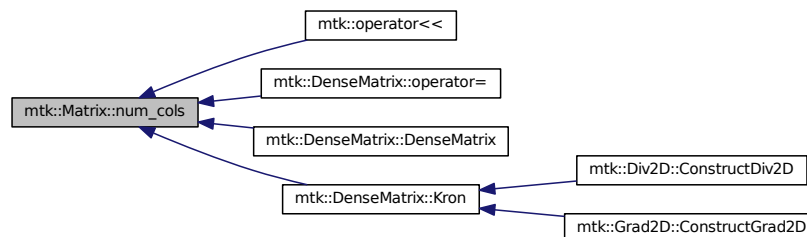
16.14.3.9 int mtk::Matrix::num\_cols ( ) const

Returns

Number of rows of the matrix.

Definition at line 127 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



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

See Also

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

Returns

Number of non-null values of the matrix.

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

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

Returns

Number of non-zero values of the matrix.

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

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

See Also

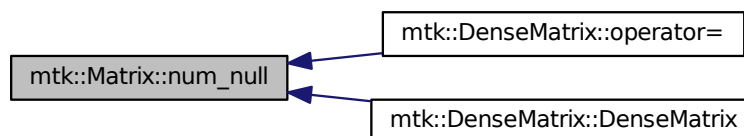
[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

Returns

Number of null values of the matrix.

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

Here is the caller graph for this function:



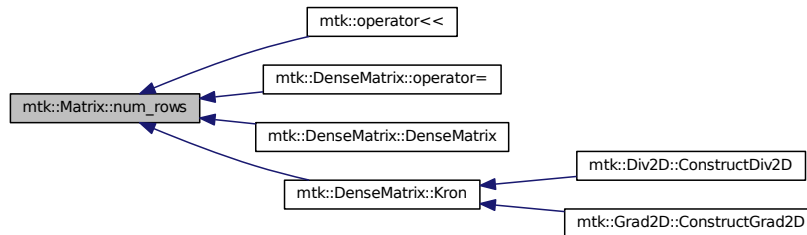
16.14.3.13 `int mtk::Matrix::num_rows ( ) const`

## Returns

Number of rows of the matrix.

Definition at line 122 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:

16.14.3.14 `int mtk::Matrix::num_values ( ) const`

## Returns

Number of values of the matrix.

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

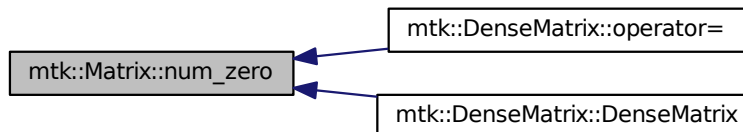
16.14.3.15 `int mtk::Matrix::num_zero ( ) const`

## Returns

Number of zeros of the matrix.

Definition at line 142 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:

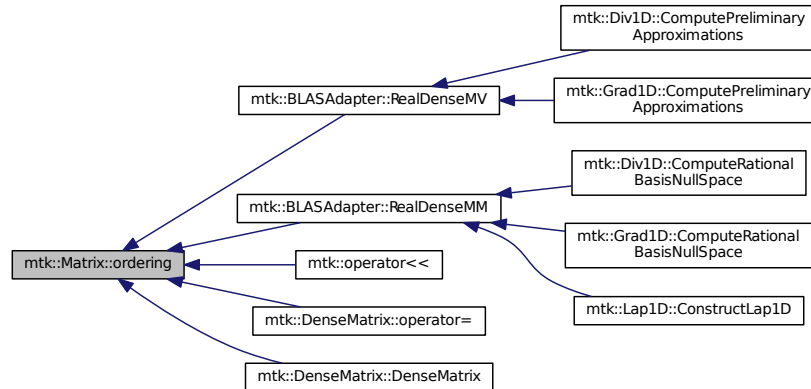
16.14.3.16 `mtk::MatrixOrdering mtk::Matrix::ordering ( ) const`

**Returns**

Type of ordering of this matrix.

Definition at line 117 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



16.14.3.17 `mtk::Real mtk::Matrix::rel_density ( ) const`

**See Also**

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

**Returns**

Relative density of the matrix.

Definition at line 177 of file [mtk\\_matrix.cc](#).

16.14.3.18 `mtk::Real mtk::Matrix::rel_sparsity ( ) const`

**See Also**

[http://www.csrc.sdsu.edu/research\\_reports/CSR2013-01.pdf](http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf)

**Returns**

Relative sparsity of the matrix.

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

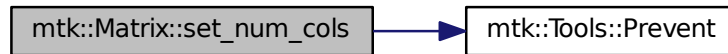
16.14.3.19 `void mtk::Matrix::set_num_cols ( int num_cols )`

## Parameters

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

Definition at line 229 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.3.20 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 255 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

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

Definition at line 217 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

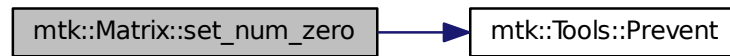
Parameters

<code>in</code>	<code><i>in</i></code>	Number of zero values.
-----------------	------------------------	------------------------

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

Definition at line 241 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.3.23 void mtk::Matrix::set\_ordering ( const MatrixOrdering & oo )

See Also

[MatrixOrdering](#)

Parameters

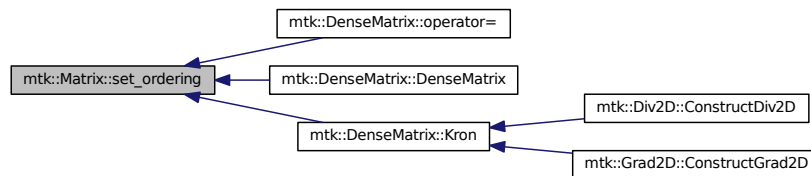
in	oo	Ordering of the matrix.
----	----	-------------------------

Definition at line 204 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.14.3.24 void mtk::Matrix::set\_storage ( const MatrixStorage & tt )

See Also

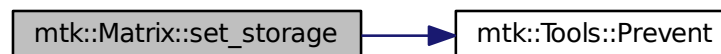
[MatrixStorage](#)

Parameters

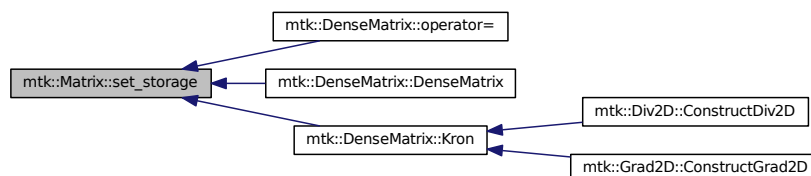
in	tt	Type of the matrix storage.
----	----	-----------------------------

Definition at line 192 of file [mtk\\_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

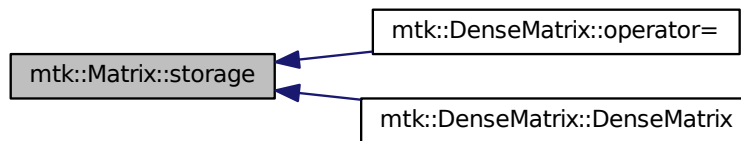


## Returns

Type of storage of this matrix.

Definition at line 112 of file [mtk\\_matrix.cc](#).

Here is the caller graph for this function:



#### 16.14.4 Member Data Documentation

##### 16.14.4.1 Real mtk::Matrix::abs\_density\_ [private]

Definition at line 296 of file [mtk\\_matrix.h](#).

##### 16.14.4.2 Real mtk::Matrix::abs\_sparsity\_ [private]

Definition at line 298 of file [mtk\\_matrix.h](#).

##### 16.14.4.3 int mtk::Matrix::bandwidth\_ [private]

Definition at line 294 of file [mtk\\_matrix.h](#).

##### 16.14.4.4 int mtk::Matrix::kl\_ [private]

Definition at line 292 of file [mtk\\_matrix.h](#).

##### 16.14.4.5 int mtk::Matrix::ku\_ [private]

Definition at line 293 of file [mtk\\_matrix.h](#).

##### 16.14.4.6 int mtk::Matrix::ld\_ [private]

Definition at line 285 of file [mtk\\_matrix.h](#).

##### 16.14.4.7 int mtk::Matrix::num\_cols\_ [private]

Definition at line 283 of file [mtk\\_matrix.h](#).

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

Definition at line 290 of file [mtk\\_matrix.h](#).

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

Definition at line 288 of file [mtk\\_matrix.h](#).

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

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

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

Definition at line 282 of file [mtk\\_matrix.h](#).

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

Definition at line 284 of file [mtk\\_matrix.h](#).

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

Definition at line 287 of file [mtk\\_matrix.h](#).

16.14.4.14 `MatrixOrdering mtk::Matrix::ordering_ [private]`

Definition at line 280 of file [mtk\\_matrix.h](#).

16.14.4.15 `Real mtk::Matrix::rel_density_ [private]`

Definition at line 297 of file [mtk\\_matrix.h](#).

16.14.4.16 `Real mtk::Matrix::rel_sparsity_ [private]`

Definition at line 299 of file [mtk\\_matrix.h](#).

16.14.4.17 `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:

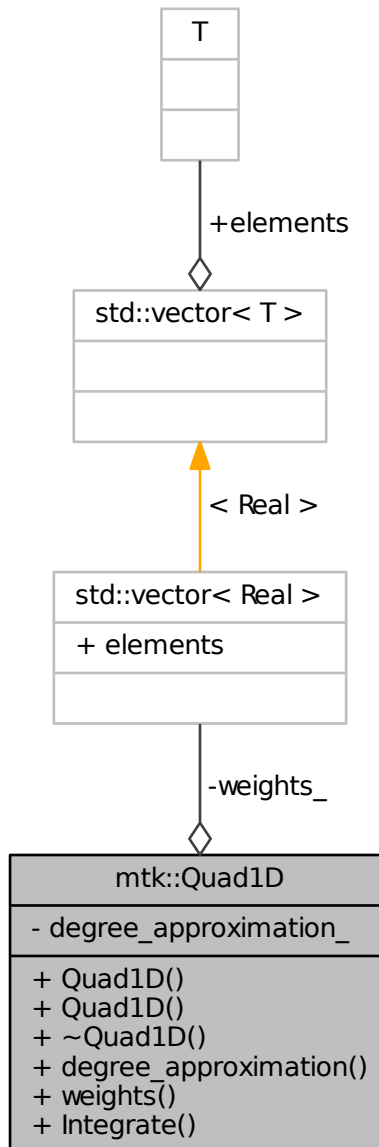
- [include/mtk\\_matrix.h](#)
- [src/mtk\\_matrix.cc](#)

## 16.15 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.*

### 16.15.1 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](#).

### 16.15.2 Constructor & Destructor Documentation

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

16.15.2.2 [mtk::Quad1D::Quad1D](#) ( const [Quad1D](#) & *quad* )

#### Parameters

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

16.15.2.3 `mtk::Quad1D::~~Quad1D ( )`

### 16.15.3 Member Function Documentation

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

#### Returns

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

16.15.3.2 `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.

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

#### Returns

Collection of weights.

### 16.15.4 Friends And Related Function Documentation

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

### 16.15.5 Member Data Documentation

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

Definition at line 124 of file [mtk\\_quad\\_1d.h](#).

16.15.5.2 `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:

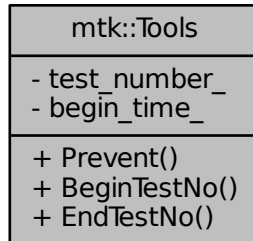
- [include/mtk\\_quad\\_1d.h](#)

## 16.16 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 condition, const char \*fname, int lineno, const char \*fxname)  
*Enforces pre-conditions by preventing their complements from occur.*
- static void [BeginTestNo](#) (const int &nn)  
*Begins the execution of a test.*
- static void [EndTestNo](#) (const int &nn)  
*Ends the execution of a test.*

### Static Private Attributes

- static int [test\\_number\\_](#)  
*Current test being executed.*
- static clock\_t [begin\\_time\\_](#)  
*Elapsed time on current test.*

#### 16.16.1 Detailed Description

Basic tools to ensure execution correctness.

Definition at line 72 of file [mtk\\_tools.h](#).

#### 16.16.2 Member Function Documentation

16.16.2.1 void mtk::Tools::BeginTestNo ( const int & *nn* ) [static]

##### Parameters

---

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

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

Here is the call graph for this function:



#### 16.16.2.2 void mtk::Tools::EndTestNo ( const int & *nn* ) [static]

##### Parameters

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

Definition at line 101 of file [mtk\\_tools.cc](#).

Here is the call graph for this function:



#### 16.16.2.3 void mtk::Tools::Prevent ( const bool *condition*, const char \* *fname*, int *lineno*, const char \* *fxname* ) [static]

##### See Also

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

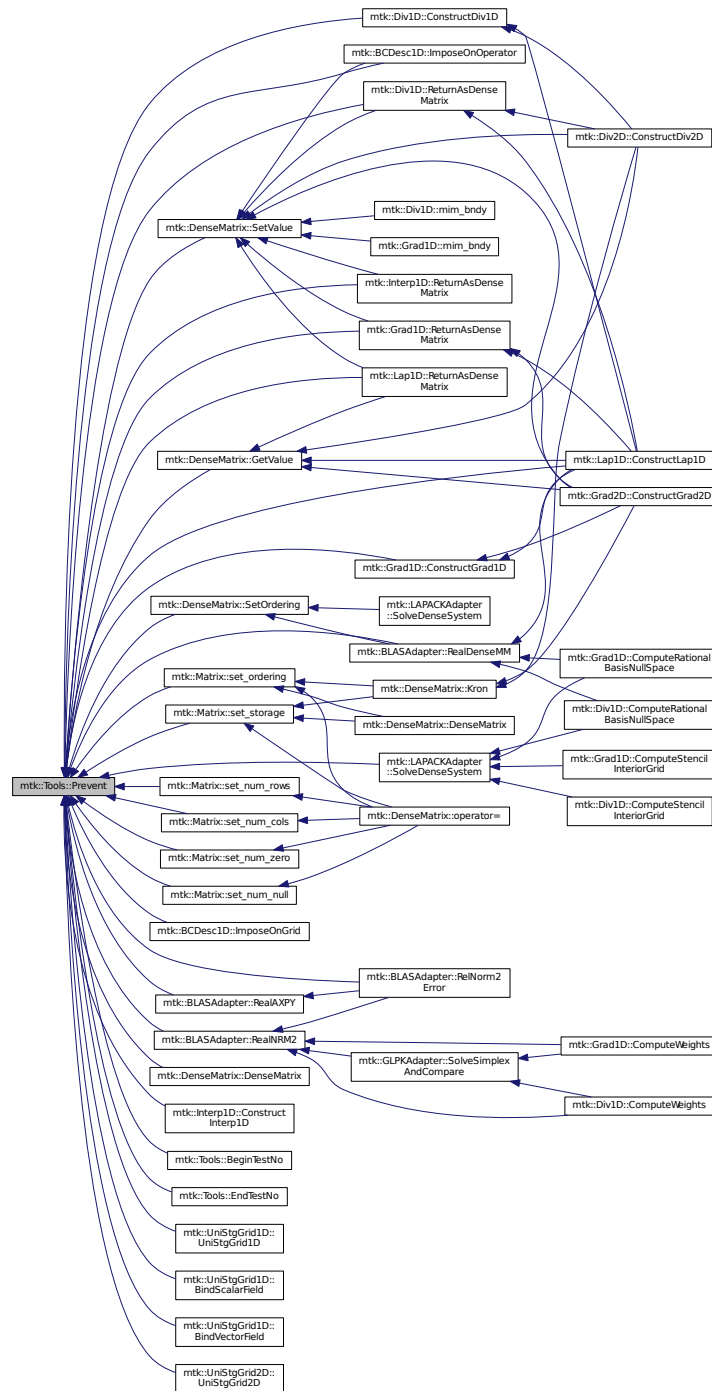
##### Parameters

<i>in</i>	<i>condition</i>	Complement of desired pre-condition.
<i>in</i>	<i>fname</i>	Name of the file being checked.
<i>in</i>	<i>lineno</i>	Number of the line where the check is executed.
<i>in</i>	<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:



### 16.16.3 Member Data Documentation



16.16.3.1 `clock_t mtk::Tools::begin_time_` `[static]`, `[private]`

Definition at line 106 of file [mtk\\_tools.h](#).

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

**Todo** Check usage of static methods and private members.

Definition at line 104 of file [mtk\\_tools.h](#).

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

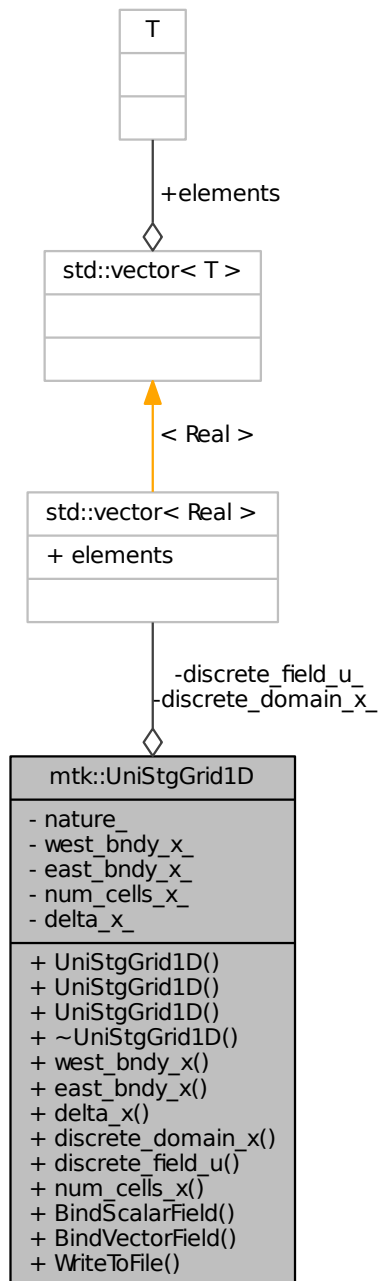
- [include/mtk\\_tools.h](#)
- [src/mtk\\_tools.cc](#)

## 16.17 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  $\Delta x$ .*

- [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  $\Delta x$ .*

## Friends

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

*Prints the grid as a tuple of arrays.*

### 16.17.1 Detailed Description

Uniform 1D Staggered Grid.

Definition at line 77 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

### 16.17.2 Constructor & Destructor Documentation

#### 16.17.2.1 `mtk::UniStgGrid1D::UniStgGrid1D ( )`

Definition at line 99 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

#### 16.17.2.2 `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](#).

#### 16.17.2.3 `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

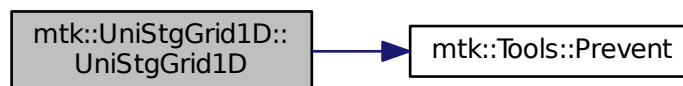
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>nature</i>	Nature of the discrete field to hold.

See Also

[mtk::FieldNature](#)

Definition at line 124 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the call graph for this function:



#### 16.17.2.4 `mtk::UniStgGrid1D::~~UniStgGrid1D ( )`

Definition at line 144 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

### 16.17.3 Member Function Documentation

#### 16.17.3.1 void mtk::UniStgGrid1D::BindScalarField ( *Real*(\*)(*Real* xx) *ScalarField* )

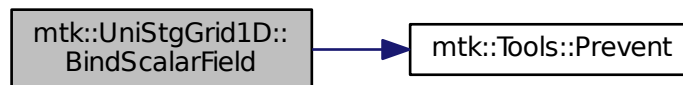
##### Parameters

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:



#### 16.17.3.2 void mtk::UniStgGrid1D::BindVectorField ( *Real*(\*)(*Real* xx) *VectorField* )

We assume the field to be of the form:

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

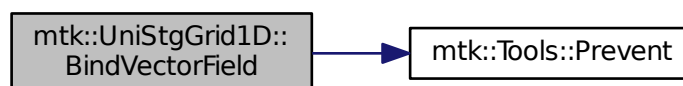
##### Parameters

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:



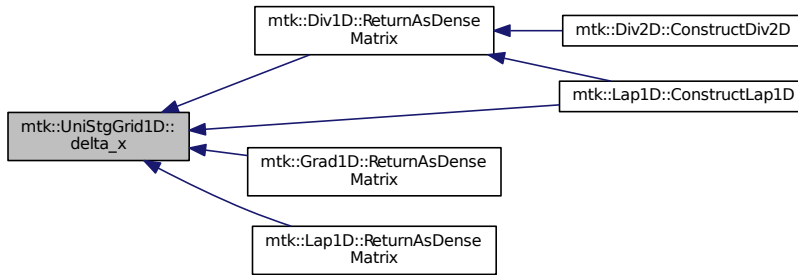
### 16.17.3.3 `mtk::Real mtk::UniStgGrid1D::delta_x ( ) const`

#### Returns

Computed  $\Delta x$ .

Definition at line 156 of file [mtk\\_uni\\_stg\\_grid\\_1d.cc](#).

Here is the caller graph for this function:



### 16.17.3.4 `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](#).

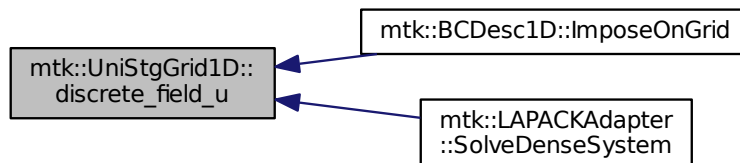
### 16.17.3.5 `mtk::Real * mtk::UniStgGrid1D::discrete_field_u ( )`

#### 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:



16.17.3.6 `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](#).

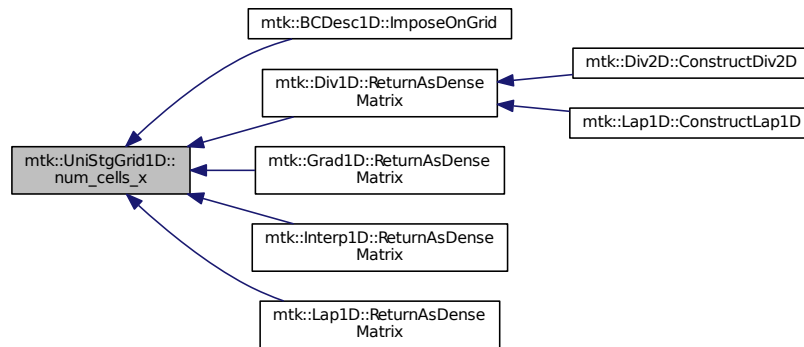
16.17.3.7 `int mtk::UniStgGrid1D::num_cells_x ( ) const`

## 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:

16.17.3.8 `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](#).

16.17.3.9 `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.

<code>in</code>	<code>field_name</code>	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](#).

**16.17.4 Friends And Related Function Documentation**

16.17.4.1 `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](#).

**16.17.5 Member Data Documentation**

16.17.5.1 `Real mtk::UniStgGrid1D::delta_x_` [*private*]

Definition at line 196 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

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

Definition at line 190 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

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

Definition at line 191 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

16.17.5.4 `Real mtk::UniStgGrid1D::east_bndy_x_` [*private*]

Definition at line 194 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

16.17.5.5 `FieldNature mtk::UniStgGrid1D::nature_` [*private*]

Definition at line 188 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).

16.17.5.6 `Real mtk::UniStgGrid1D::num_cells_x_` [*private*]

Definition at line 195 of file [mtk\\_uni\\_stg\\_grid\\_1d.h](#).



16.17.5.7 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](#)

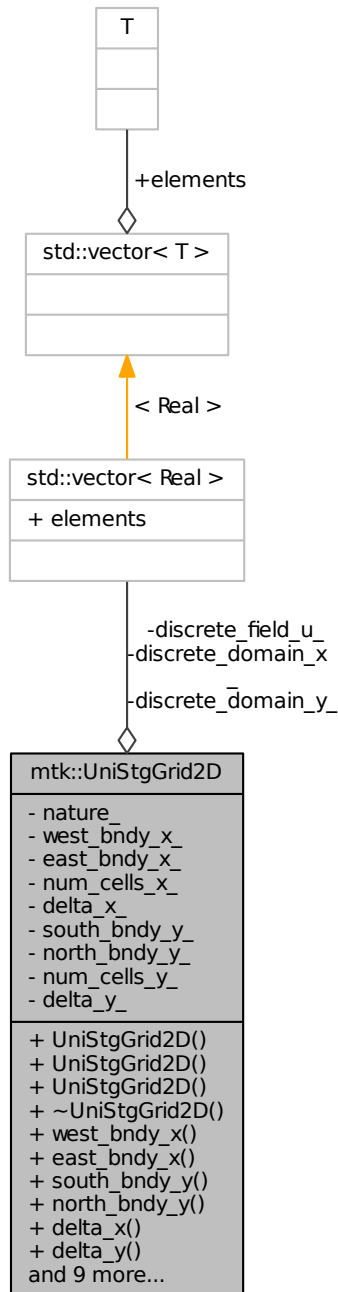
- [src/mtk\\_uni\\_stg\\_grid\\_1d.cc](#)

## 16.18 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::SCALAR](#))

*Construct a grid based on spatial discretization parameters.*

- [~UniStgGrid2D](#) ()

*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](#) south\_bndy\_y () const

*Provides access to south boundary spatial coordinate.*

- [Real](#) north\_bndy\_y () const

*Provides access to north boundary spatial coordinate.*

- [Real](#) delta\_x () const

*Provides access to the computed  $\Delta x$ .*

- [Real](#) delta\_y () const

*Provides access to the computed  $\Delta y$ .*

- [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\_u ()

*Provides access to the grid field data.*

- int num\_cells\_x () const

*Provides access to the number of cells of the grid.*

- int num\_cells\_y () const

*Provides access to the number of cells of the grid.*

- void [BindScalarField](#) ([Real](#)(\*ScalarField)([Real](#) xx, [Real](#) yy))

*Binds a given scalar field to the grid.*

- void [BindVectorFieldPComponent](#) ([Real](#)(\*VectorField)([Real](#) xx, [Real](#) yy))

*Binds a given vector field to the grid.*

- void [BindVectorFieldQComponent](#) ([Real](#)(\*VectorField)([Real](#) xx, [Real](#) yy))

*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\_domain\_y\_

*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  $\Delta x$ .*
- `Real south_bndy_y_`  
*West boundary spatial coordinate.*
- `Real north_bndy_y_`  
*East boundary spatial coordinate.*
- `Real num_cells_y_`  
*Number of cells discretizing the domain.*
- `Real delta_y_`  
*Produced  $\Delta y$ .*

## Friends

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

### 16.18.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 79 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

### 16.18.2 Constructor & Destructor Documentation

16.18.2.1 `mtk::UniStgGrid2D::UniStgGrid2D ( )`

Definition at line 111 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

16.18.2.2 `mtk::UniStgGrid2D::UniStgGrid2D ( const UniStgGrid2D &grid )`

#### Parameters

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

Definition at line 125 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

16.18.2.3 `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

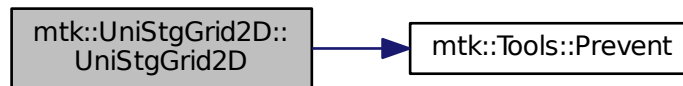
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</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 149 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the call graph for this function:



## 16.18.2.4 mtk::UniStgGrid2D::~~UniStgGrid2D ( )

Definition at line 183 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

## 16.18.3 Member Function Documentation

16.18.3.1 void mtk::UniStgGrid2D::BindScalarField ( Real(\*) (Real xx, Real yy) *ScalarField* )

## Parameters

in	<i>ScalarField</i>	Pointer to the function implementing the scalar field.
----	--------------------	--

16.18.3.2 void mtk::UniStgGrid2D::BindVectorFieldPComponent ( Real(\*) (Real xx, Real yy) *VectorField* )

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>VectorField</i>	Pointer to the function implementing the vector field.
----	--------------------	--

16.18.3.3 void mtk::UniStgGrid2D::BindVectorFieldQComponent ( Real(\*) (Real xx, Real yy) *VectorField* )

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>VectorField</i>	Pointer to the function implementing the vector field.
----	--------------------	--

16.18.3.4 Real mtk::UniStgGrid2D::delta\_x ( ) const

## Returns

Computed  $\Delta x$ .

16.18.3.5 Real mtk::UniStgGrid2D::delta\_y ( ) const

## Returns

Computed  $\Delta y$ .

16.18.3.6 Real\* mtk::UniStgGrid2D::discrete\_domain\_x ( )

## Returns

Pointer to the spatial data.

16.18.3.7 Real\* mtk::UniStgGrid2D::discrete\_domain\_y ( )

## Returns

Pointer to the spatial data.

16.18.3.8 Real\* mtk::UniStgGrid2D::discrete\_field\_u ( )

## Returns

Pointer to the field data.

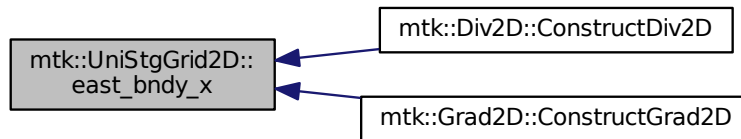
### 16.18.3.9 mtk::Real mtk::UniStgGrid2D::east\_bndy\_x ( ) const

#### Returns

East boundary spatial coordinate.

Definition at line 190 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



### 16.18.3.10 mtk::Real mtk::UniStgGrid2D::north\_bndy\_y ( ) const

#### Returns

North boundary spatial coordinate.

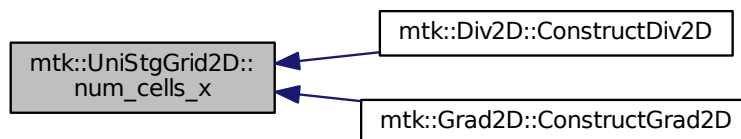
Definition at line 200 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

### 16.18.3.11 int mtk::UniStgGrid2D::num\_cells\_x ( ) const

#### Returns

Number of cells of the grid.

Here is the caller graph for this function:

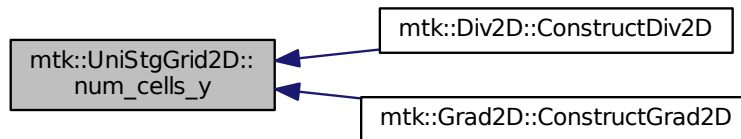


### 16.18.3.12 int mtk::UniStgGrid2D::num\_cells\_y ( ) const

**Returns**

Number of cells of the grid.

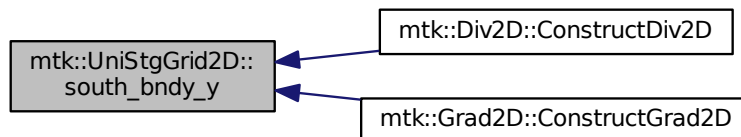
Here is the caller graph for this function:

**16.18.3.13** `mtk::Real mtk::UniStgGrid2D::south_bndy_y ( ) const`**Returns**

South boundary spatial coordinate.

Definition at line [195](#) of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:

**16.18.3.14** `mtk::Real mtk::UniStgGrid2D::west_bndy_x ( ) const`

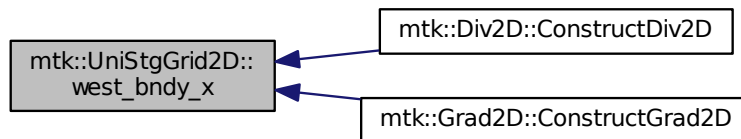


**Returns**

West boundary spatial coordinate.

Definition at line 185 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

Here is the caller graph for this function:



**16.18.3.15** `bool mtk::UniStgGrid2D::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/>

**16.18.4 Friends And Related Function Documentation**

**16.18.4.1** `std::ostream& operator<< ( std::ostream & stream, mtk::UniStgGrid2D & in )` [*friend*]

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

Definition at line 66 of file [mtk\\_uni\\_stg\\_grid\\_2d.cc](#).

**16.18.5 Member Data Documentation**

**16.18.5.1** `Real mtk::UniStgGrid2D::delta_x_` [*private*]

Definition at line 253 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.2** `Real mtk::UniStgGrid2D::delta_y_` [private]

Definition at line 258 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

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

Definition at line 246 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

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

Definition at line 247 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.5** `std::vector<Real> mtk::UniStgGrid2D::discrete_field_u_` [private]

Definition at line 248 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.6** `Real mtk::UniStgGrid2D::east_bndy_x_` [private]

Definition at line 251 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.7** `FieldNature mtk::UniStgGrid2D::nature_` [private]

Definition at line 244 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.8** `Real mtk::UniStgGrid2D::north_bndy_y_` [private]

Definition at line 256 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.9** `Real mtk::UniStgGrid2D::num_cells_x_` [private]

Definition at line 252 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.10** `Real mtk::UniStgGrid2D::num_cells_y_` [private]

Definition at line 257 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.11** `Real mtk::UniStgGrid2D::south_bndy_y_` [private]

Definition at line 255 of file [mtk\\_uni\\_stg\\_grid\\_2d.h](#).

**16.18.5.12** `Real mtk::UniStgGrid2D::west_bndy_x_` [private]

Definition at line 250 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

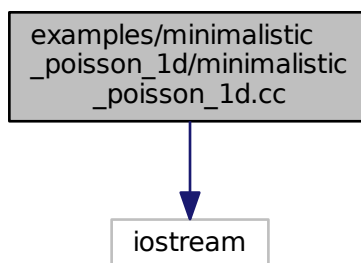
# File Documentation

### 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 183 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. Documentation related to said modifications should be included.
00052
00053 2. Redistributions of source code must be done through direct
00054 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00055
00056 3. Redistributions of source code must retain the above copyright notice, this
00057 list of conditions and the following disclaimer.
00058
00059 4. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 5. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders.
```

```

00065
00066 6. 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     mtk::Real west_bndy_x = 0.0;
00115     mtk::Real east_bndy_x = 1.0;
00116     mtk::Real relative_norm_2_error{};
00117     int num_cells_x = 5;
00118     mtk::Grad1D grad;
00119     mtk::Lapl1D lap;
00120     std::vector<mtk::Real> west_coeffs;
00121     std::vector<mtk::Real> east_coeffs;
00122     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00123     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00124     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00125
00126     if (!lap.ConstructLapl1D()) {
00127         std::cerr << "Mimetic lap could not be built." << std::endl;
00128         return EXIT_FAILURE;
00129     }
00130
00131     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00132
00133     if (!grad.ConstructGrad1D()) {
00134         std::cerr << "Mimetic grad could not be built." << std::endl;
00135         return EXIT_FAILURE;
00136     }
00137
00138     mtk::DenseMatrix gradm(grad.ReturnAsDenseMatrix(comp_sol));
00139
00140     source.BindScalarField(Source);
00141
00142     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00143         west_coeffs.push_back(-(exp(-1.0) - 1.0)/-1.0)*gradm.GetValue(0, ii);
00144     }
00145

```

```

00146     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00147         east_coeffs.push_back(((exp(-1.0) - 1.0)/-1.0)*gradm.GetValue(gradm.num_rows() - 1,
00148                                 gradm.num_cols() - 1 - ii));
00149     }
00150
00151     west_coeffs[0] += -exp(-1.0);
00152
00153     east_coeffs[0] += -exp(-1.0);
00154
00155     mtk::BCDesc1D::ImposeOnOperator(lapm, west_coeffs, east_coeffs);
00156
00157     mtk::BCDesc1D::ImposeOnGrid(source, -1.0, 0.0);
00158
00159     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00160
00161     if (info != 0) {
00162         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00163         return EXIT_FAILURE;
00164     }
00165
00166     source.WriteToFile("minimalistic_poisson_1d_comp_sol.dat", "x", "~u(x)");
00167
00168     known_sol.BindScalarField(KnownSolution);
00169
00170     relative_norm_2_error =
00171         mtk::BLASAdapter::RelNorm2Error(source.discrete_field_u(),
00172                                         known_sol.discrete_field_u(),
00173                                         known_sol.num_cells_x());
00174
00175     std::cout << "relative_norm_2_error = ";
00176     std::cout << relative_norm_2_error << std::endl;
00177 }
00178
00179 #else
00180 #include <iostream>
00181 using std::cout;
00182 using std::endl;
00183 int main () {
00184     cout << "This code HAS to be compiled with support for C++11." << endl;
00185     cout << "Exiting..." << endl;
00186     return EXIT_SUCCESS;
00187 }
00188 #endif

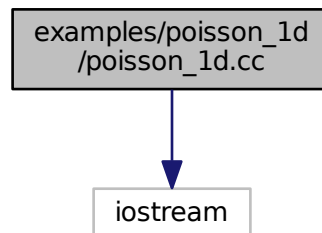
```

### 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. Documentation related to said modifications should be included.
00052
00053 2. Redistributions of source code must be done through direct
00054 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00055
00056 3. Redistributions of source code must retain the above copyright notice, this
00057 list of conditions and the following disclaimer.
00058
00059 4. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 5. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders.
00065
00066 6. 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
00118     mtk::Real lambda = -1.0;
00119     mtk::Real alpha = -exp(lambda);
00120     mtk::Real beta = (exp(lambda) - 1.0)/lambda;
00121     mtk::Real omega = -1.0;
00122     mtk::Real epsilon = 0.0;
00123
00124
00125     mtk::Real west_bndy_x = 0.0;
00126     mtk::Real east_bndy_x = 1.0;

```

```

00129     int num_cells_x = 5;
00130
00131     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00132
00133     int order_of_accuracy{2}; // Desired order of accuracy for approximation.
00134
00135     mtk::Grad1D grad; // Mimetic gradient operator.
00136
00137     mtk::Lapl1D lap; // Mimetic Laplacian operator.
00138
00139     if (!lap.ConstructLapl1D(order_of_accuracy)) {
00140         std::cerr << "Mimetic lap could not be built." << std::endl;
00141         return EXIT_FAILURE;
00142     }
00143
00144     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00145
00146     std::cout << "Mimetic Laplacian operator: " << std::endl;
00147     std::cout << lapm << std::endl;
00148
00149     if (!grad.ConstructGrad1D(order_of_accuracy)) {
00150         std::cerr << "Mimetic grad could not be built." << std::endl;
00151         return EXIT_FAILURE;
00152     }
00153
00154     mtk::DenseMatrix gradm(grad.ReturnAsDenseMatrix(comp_sol));
00155
00156     std::cout << "Mimetic gradient operator: " << std::endl;
00157     std::cout << gradm << std::endl;
00158
00159     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00160
00161     source.BindScalarField(Source);
00162
00163     std::cout << source << std::endl;
00164
00165     // Since we need to approximate the first derivative times beta, we must use
00166     // the approximation of the gradient at the boundary. We could extract them
00167     // from the gradient operator as packed in the grad object. BUT, since we have
00168     // generated at matrix containing this operator, we can extract these from the
00169     // matrix.
00170
00171     // Array containing the coefficients for the west boundary condition.
00172     std::vector<mtk::Real> west_coeffs;
00173
00174     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00175         west_coeffs.push_back(-beta*gradm.GetValue(0, ii));
00176     }
00177
00178     // Array containing the coefficients for the east boundary condition.
00179     std::vector<mtk::Real> east_coeffs;
00180
00181     for (auto ii = 0; ii < grad.num_bndy_coeffs(); ++ii) {
00182         east_coeffs.push_back(beta*gradm.GetValue(gradm.num_rows() - 1,
00183                                                    gradm.num_cols() - 1 - ii));
00184     }
00185
00186     // To impose the Dirichlet condition, we simple add its coefficient to the
00187     // first entry of the west, and the last entry of the east array.
00188
00189     west_coeffs[0] += alpha;
00190
00191     east_coeffs[0] += alpha;
00192
00193     // Now that we have the coefficients that should be in the operator, we create
00194     // a boundary condition descriptor object, which will encapsulate the
00195     // complexity of assigning them in the matrix, to complete the construction of
00196     // the mimetic operator.
00197
00198     mtk::BCDesc1D::ImposeOnOperator(lapm, west_coeffs, east_coeffs);
00199
00200     std::cout << "Mimetic Laplacian with Robin conditions:" << std::endl;
00201     std::cout << lapm << std::endl;
00202
00203     mtk::BCDesc1D::ImposeOnGrid(source, omega, epsilon);
00204
00205     std::cout << "Source term with imposed BCs:" << std::endl;
00206     std::cout << source << std::endl;

```

```

00213
00214     source.WriteToFile("poisson_1d_source.dat", "x", "s(x)");
00215
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
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

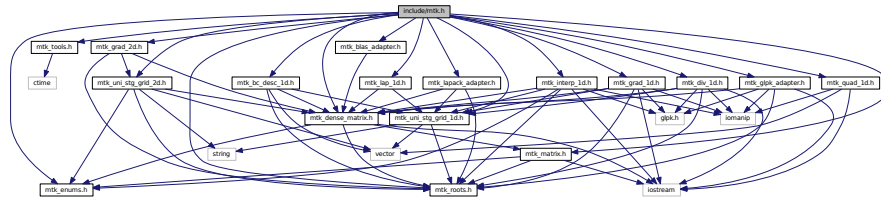
Includes the entire API.

```

#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"

```

Include dependency graph for mtk.h:



### 17.5.1 Detailed Description

This file contains every required header file, thus containing the entire API. In this way, client codes only have to instruct `#include "mtk.h"`.

#### Warning

IT IS EXTREMELY IMPORTANT THAT THE HEADERS ARE ADDED TO THIS FILE IN A SPECIFIC ORDER; THAT IS, CONSIDERING THE DEPENDENCE BETWEEN THE CLASSES THESE CONTAIN!

#### Author

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

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

## 17.6 mtk.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:

```

```

00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. 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 */
00362 #ifndef MTK_INCLUDE_MTK_H_
00363 #define MTK_INCLUDE_MTK_H_
00364
00372 #include "mtk_roots.h"
00373
00381 #include "mtk_enums.h"
00382
00390 #include "mtk_tools.h"
00391
00399 #include "mtk_matrix.h"
00400 #include "mtk_dense_matrix.h"
00401
00409 #include "mtk_blas_adapter.h"
00410 #include "mtk_lapack_adapter.h"
00411 #include "mtk_glpk_adapter.h"
00412
00420 #include "mtk_uni_stg_grid_1d.h"
00421 #include "mtk_uni_stg_grid_2d.h"
00422
00430 #include "mtk_grad_1d.h"
00431 #include "mtk_div_1d.h"
00432 #include "mtk_lap_1d.h"
00433 #include "mtk_interp_1d.h"
00434 #include "mtk_quad_1d.h"
00435 #include "mtk_bc_desc_1d.h"
00436
00437 #include "mtk_grad_2d.h"
00438
00439 #endif // End of: MTK_INCLUDE_MTK_H_

```

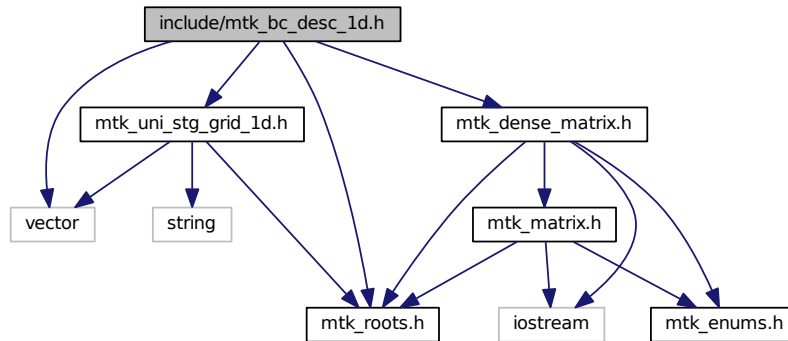
## 17.7 include/mtk\_bc\_desc\_1d.h File Reference

```

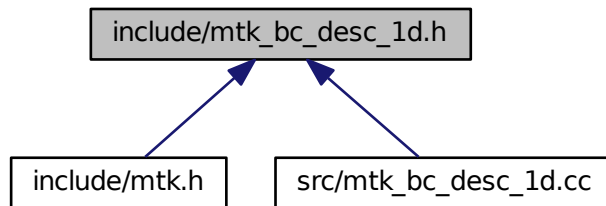
#include <vector>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.*

## 17.8 mtk\_bc\_desc\_1d.h

```

00001 #include <vector>
00002
00003 #include "mtk_roots.h"
00004 #include "mtk_dense_matrix.h"
00005 #include "mtk_uni_stg_grid_1d.h"
  
```

```

00006
00007 namespace mtk {
00008
00009 class BCDesc1D {
00010 public:
00011     static void ImposeOnOperator(DenseMatrix &matrix,
00012                                 const std::vector<Real> &west,
00013                                 const std::vector<Real> &east);
00014
00015     static void ImposeOnGrid(UniStgGrid1D &grid,
00016                             const Real &omega,
00017                             const Real &epsilon);
00018 };
00019 }

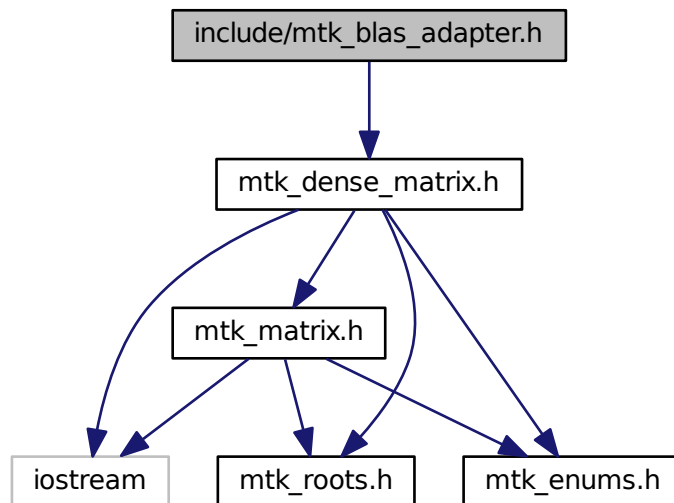
```

## 17.9 include/mtk\_blas\_adapter.h File Reference

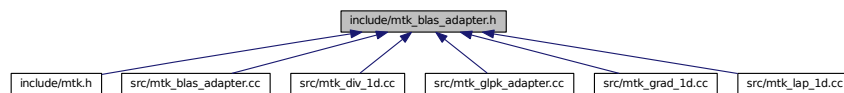
Adapter class for the BLAS API.

```
#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.*

### 17.9.1 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](#).

## 17.10 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. Documentation related to said modifications should be included.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions of source code must retain the above copyright notice, this
00039 list of conditions and the following disclaimer.
00040
00041 4. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 5. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders.
00047
00048 6. 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_

```

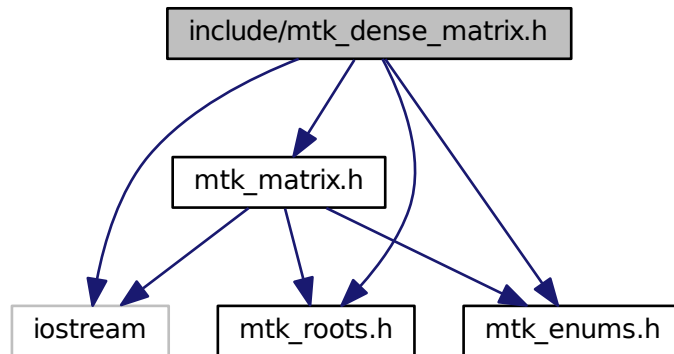
## 17.11 include/mtk\_dense\_matrix.h File Reference

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

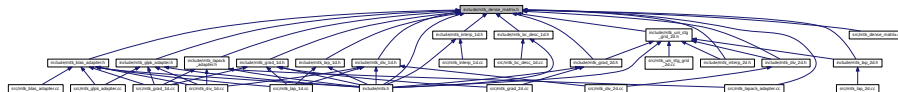
```

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

```



**Included:** those studies



- class `mtk::DenseMatrix`

## Namespaces

- Mimetic Methods Toolkit namespace.*

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.

: 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](#).

## 17.12 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00038 list of conditions and the following disclaimer.
00039
00040 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00045 prior written permission from the the copyright holders.
00046
00047 6. 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     DenseMatrix();
00102
00108     DenseMatrix(const DenseMatrix &in);
00109
00118     DenseMatrix(const int &num_rows, const int &num_cols);
00119
00145     DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00146
00180     DenseMatrix(const Real *gen,
00181                 const int &gen_length,
00182                 const int &pro_length,
00183                 const bool &transpose);
00184
00186     ~DenseMatrix();
00187
00193     Matrix matrix_properties() const;
00194
00200     int num_rows() const;
00201
00207     int num_cols() const;
00208
00214     Real* data() const;
00215
00223     void SetOrdering(mtk::MatrixOrdering oo);
00224
00233     Real GetValue(const int &row_coord, const int &col_coord) const;
00234
00242     void SetValue(const int &row_coord,
00243                  const int &col_coord,
00244                  const Real &val);
00245
00247     void Transpose();
00248
00250     void OrderRowMajor();
00251
00253     void OrderColMajor();
00254
00265     static DenseMatrix Kron(const DenseMatrix &aa, const
DenseMatrix &bb);
00266
00267     private:
00268     Matrix matrix_properties_;
00269
00270     Real *data_;
00271 };
00272 }
00273 #endif // End of: MTK_INCLUDE_MTK_DENSE_MATRIX_H_

```

## 17.13 include/mtk\_div\_1d.h File Reference

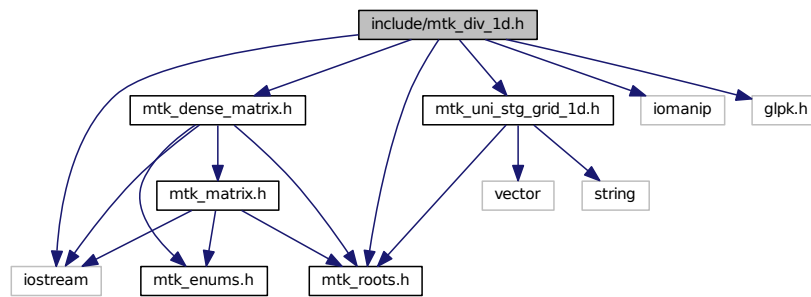
Includes the definition of the class Div1D.

```

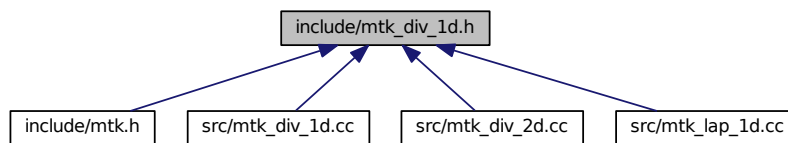
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.*

### 17.13.1 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](#).

## 17.14 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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:
00083     friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00084
00085     Div1D();
00086
00087     Div1D(const Div1D &div);
00088
00089     ~Div1D();
00090
00091     bool ConstructDiv1D(int order_accuracy = kDefaultOrderAccuracy,
00092                        Real mimetic_threshold = kDefaultMimeticThreshold);
00093
00094     int num_bndy_coeffs() const;
00095
00096     Real *coeffs_interior() const;
00097
00098 };
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
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_

```

## 17.15 include/mtk\_div\_2d.h File Reference

Includes the definition of the class Div2D.

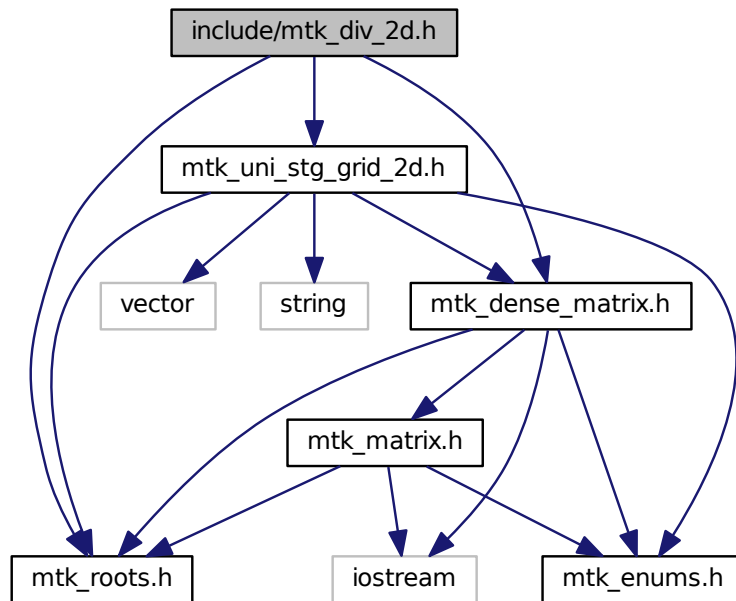
```

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

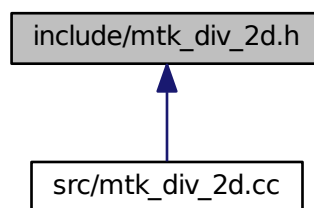
```



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.*

### 17.15.1 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](#).

## 17.16 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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{

```

```

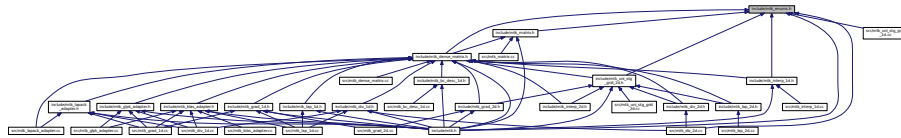
00065
00066 class Div2D {
00067 public:
00068     Div2D();
00070
00076     Div2D(const Div2D &div);
00077
00079     ~Div2D();
00080
00086     DenseMatrix ConstructDiv2D(const UniStgGrid2D &grid,
00087                             int order_accuracy = kDefaultOrderAccuracy,
00088                             Real mimetic_threshold =
00089                             kDefaultMimeticThreshold);
00095     DenseMatrix ReturnAsDenseMatrix();
00096
00097 private:
00098     DenseMatrix divergence_;
00099     int order_accuracy_;
00100     Real mimetic_threshold_;
00101 };
00102
00103 #endif // End of: MTK_INCLUDE_MTK_DIV_2D_H_

```

## 17.17 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.*

#### 17.17.1 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](#).

**17.18 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. 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_

```

## 17.19 include/mtk\_glpk\_adapter.h File Reference

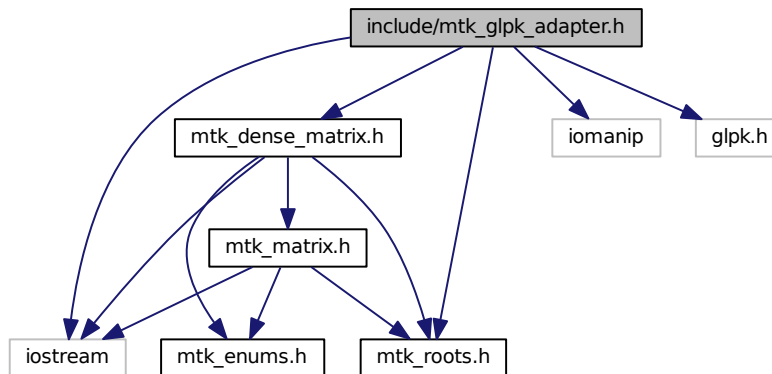
Adapter class for the GLPK API.

```

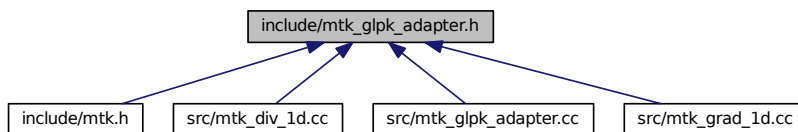
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"

```

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.*

### 17.19.1 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](#).

## 17.20 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00034 list of conditions and the following disclaimer.
00035
00036 4. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 5. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders.
00042
00043 6. 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 nrow,
00126         int ncol,
00127         mtk::Real *hh,
00128         mtk::Real *qq,
00129         int robjective,
00130         mtk::Real mimetic_tol,
00131         int copy);
00132 };
00133 }
00134 #endif // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_

```

## 17.21 include/mtk\_grad\_1d.h File Reference

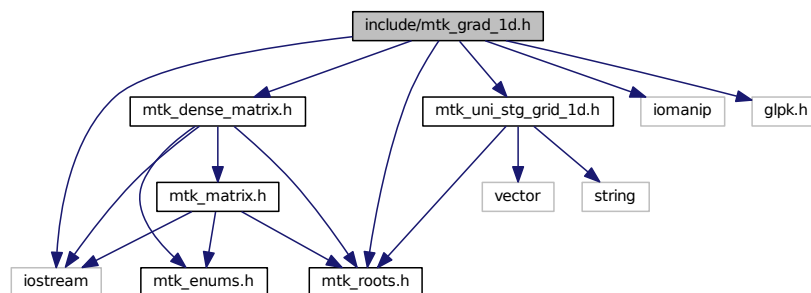
Includes the definition of the class Grad1D.

```

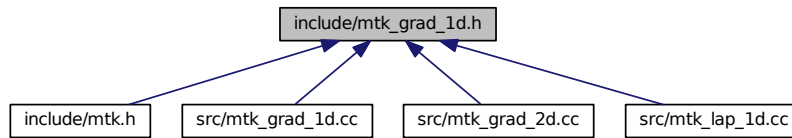
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.*

### 17.21.1 Detailed Description

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

#### Author

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

Definition in file [mtk\\_grad\\_1d.h](#).

## 17.22 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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:
00083     friend std::ostream& operator <<(std::ostream& stream, Grad1D &in);
00084
00085     Grad1D();
00086
00087     Grad1D(const Grad1D &grad);
00088
00089     ~Grad1D();
00090
00091     bool ConstructGrad1D(int order_accuracy = kDefaultOrderAccuracy,
00092                         Real mimetic_threshold = kDefaultMimeticThreshold);
00093
00094     int num_bndy_coeffs() const;
00095
00096     Real *coeffs_interior() const;
00097
00098     Real *weights_crs(void) const;
00099
00100     Real *weights_cbs(void) const;
00101
00102     DenseMatrix mim_bndy() const;
00103
00104     DenseMatrix ReturnAsDenseMatrix(Real west,
00105                                     Real east, int num_cells_x);
00106
00107     DenseMatrix ReturnAsDenseMatrix(const
00108                                     UniStgGrid1D &grid);
00109
00110 private:
00111     bool ComputeStencilInteriorGrid(void);
00112
00113     bool ComputeRationalBasisNullSpace(void);
00114
00115     bool ComputePreliminaryApproximations(void);
00116
00117     bool ComputeWeights(void);
00118
00119     bool ComputeStencilBoundaryGrid(void);
00120
00121     bool AssembleOperator(void);

```

```

00199
00200     int order_accuracy_;
00201     int dim_null_;
00202     int num_bndy_approxs_;
00203     int num_bndy_coeffs_;
00204     int gradient_length_;
00205     int minrow_;
00206     int row_;
00207
00208     DenseMatrix rat_basis_null_space_;
00209
00210     Real *coeffs_interior_;
00211     Real *prem_apps_;
00212     Real *weights_crs_;
00213     Real *weights_cbs_;
00214     Real *mim_bndy_;
00215     Real *gradient_;
00216
00217     Real mimetic_threshold_;
00218 };
00219 }
00220 #endif // End of: MTK_INCLUDE_GRAD_1D_H_

```

## 17.23 include/mtk\_grad\_2d.h File Reference

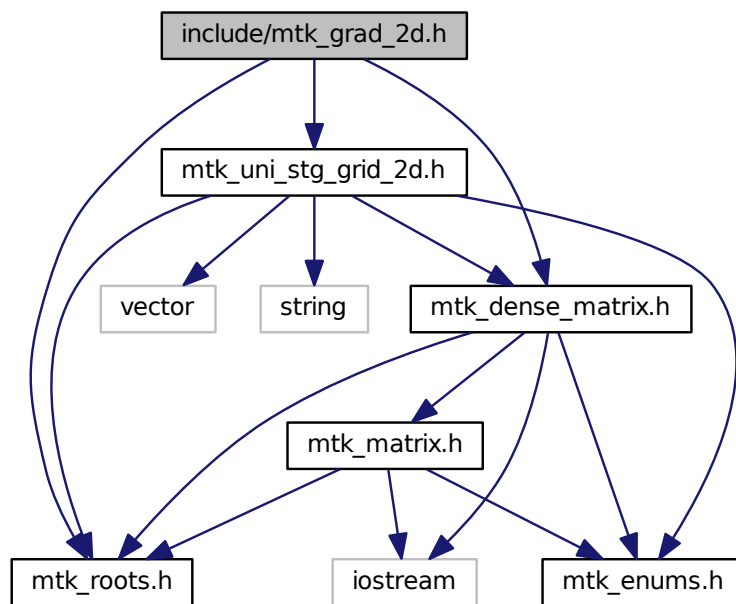
Includes the definition of the class Grad2D.

```

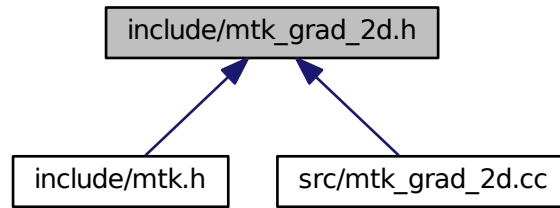
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

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.*

### 17.23.1 Detailed Description

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

#### Author

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

Definition in file [mtk\\_grad\\_2d.h](#).

## 17.24 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
  
```

```

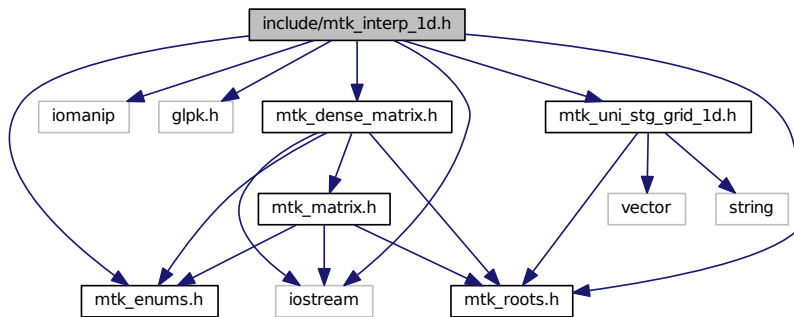
00026 list of conditions and the following disclaimer.
00027
00028 4. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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();
00069
00070     Grad2D(const Grad2D &grad);
00071
00072     ~Grad2D();
00073
00074     DenseMatrix ConstructGrad2D(const UniStgGrid2D &grid,
00075                               int order_accuracy = kDefaultOrderAccuracy,
00076                               Real mimetic_threshold =
00077                                   kDefaultMimeticThreshold);
00078
00079     DenseMatrix ReturnAsDenseMatrix();
00080
00081 private:
00082     DenseMatrix gradient_;
00083     int order_accuracy_;
00084     Real mimetic_threshold_;
00085 };
00086
00087 #endif // End of: MTK_INCLUDE_MTK_GRAD_2D_H_

```

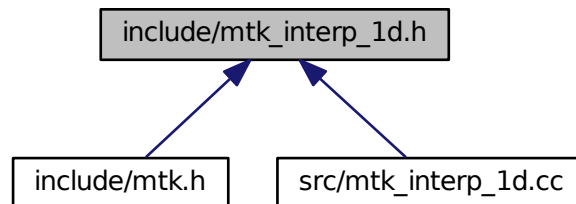
## 17.25 include/mtk\_interp\_1d.h File Reference

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.*

### 17.25.1 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](#).

## 17.26 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. 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_

```

## 17.27 include/mtk\_interp\_2d.h File Reference

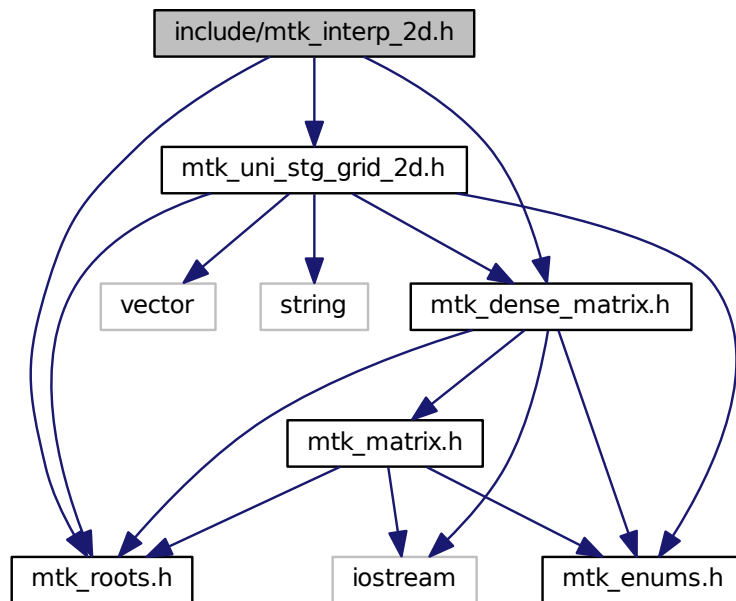
Includes the definition of the class Interp2D.

```

#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.*

### 17.27.1 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](#).



## 17.28 mtk\_interp\_2d.h

```

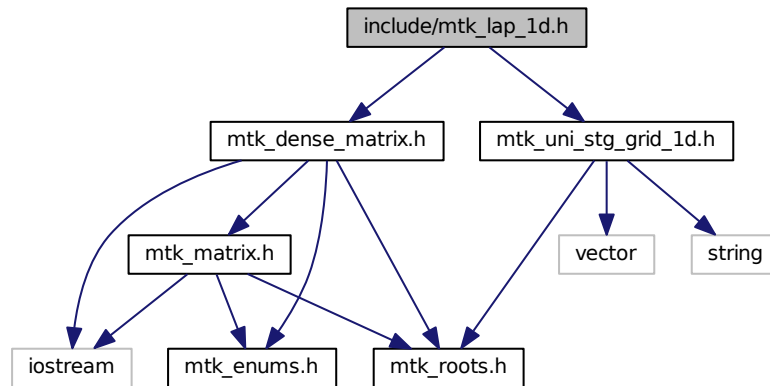
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. 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     Interp2D(const Interp2D &interp);
00071     ~Interp2D();
00072     DenseMatrix ConstructInterp2D(const UniStgGrid2D &grid,
00073                                 int order_accuracy = kDefaultOrderAccuracy,
00074                                 Real mimetic_threshold =
00075                                     kDefaultMimeticThreshold);
00076     DenseMatrix ReturnAsDenseMatrix();
00077 private:
00078     DenseMatrix interpolator_;
00079     int order_accuracy_;
00080     Real mimetic_threshold_;
00081 };
00082 }
00083 #endif // End of: MTK_INCLUDE_MTK_INTERP_2D_H_

```

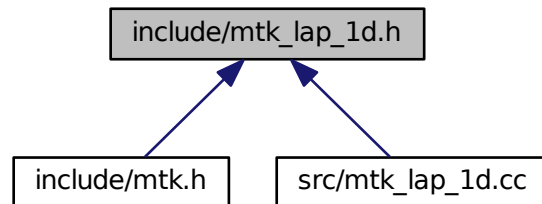
## 17.29 include/mtk\_lap\_1d.h File Reference

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.*

### 17.29.1 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](#).

## 17.30 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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:
00079     friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00080
00082     Lap1D();
00083
00089     Lap1D(const Lap1D &lap);
00090
00092     ~Lap1D();
00093
00099     bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00100                         Real mimetic_threshold = kDefaultMimeticThreshold);
00101
00107     DenseMatrix ReturnAsDenseMatrix(const
00108     UniStgGrid1D &grid);
00108
00114     mtk::Real* Data(const UniStgGrid1D &grid);
00115
00116 private:
00117     int order_accuracy_;
00118     int laplacian_length_;
00119
00120     Real *laplacian_;
00121
00122     Real mimetic_threshold_;
00123 };
00124 }
00125 #endif // End of: MTK_INCLUDE_LAP_1D_H_

```

## 17.31 include/mtk\_lap\_2d.h File Reference

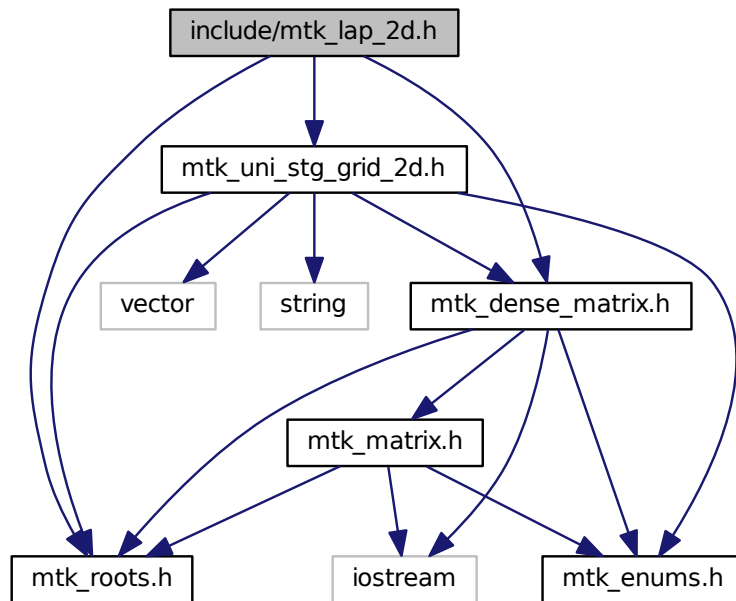
Includes the implementation of the class Lap2D.

```

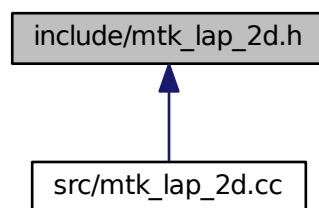
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

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.*

### 17.31.1 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](#).

## 17.32 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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     Lap2D(const Lap2D &lap);
00070     ~Lap2D();
00071     DenseMatrix ConstructLap2D(const UniStgGrid2D &grid,
00072                               int order_accuracy = kDefaultOrderAccuracy,
00073                               Real mimetic_threshold =
00074                               kDefaultMimeticThreshold);
00074     DenseMatrix ReturnAsDenseMatrix();
00075
00076 private:
00077     DenseMatrix laplacian_;
00078     int order_accuracy_;
00079     Real mimetic_threshold_;
00080 };
00081 }
00082 #endif // End of: MTK_INCLUDE_MTK_LAP_2D_H_

```

## 17.33 include/mtk\_lapack\_adapter.h File Reference

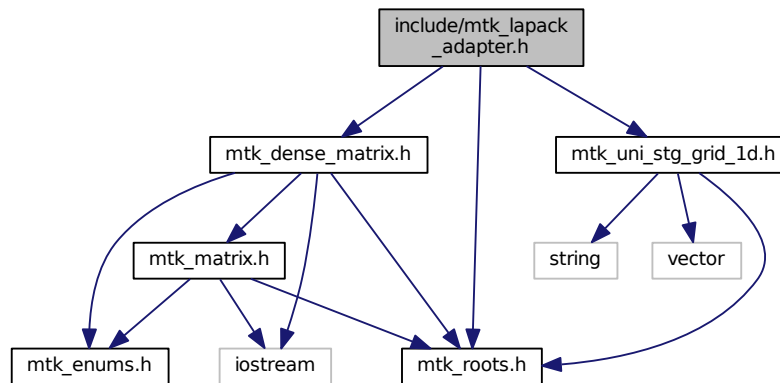
Adapter class for the LAPACK API.

```

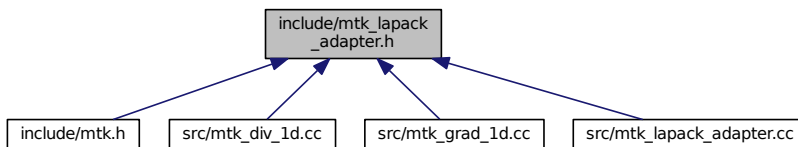
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.*

### 17.33.1 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`.

## 17.34 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00034 list of conditions and the following disclaimer.
00035
00036 4. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 5. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders.
00042
00043 6. 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_id.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::UniStgGrid1D &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_

```

## 17.35 include/mtk\_matrix.h File Reference

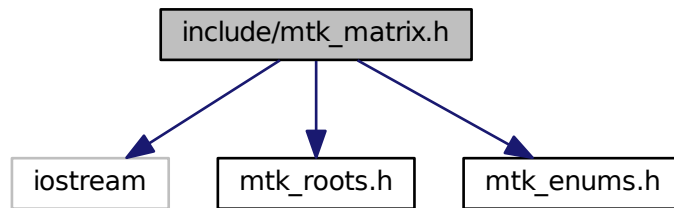
Definition of the representation of a matrix in the MTK.

```

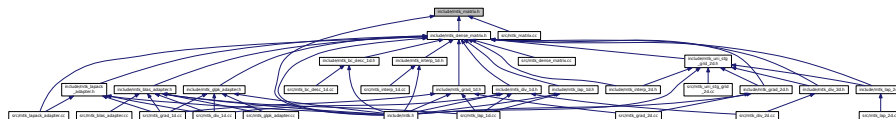
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"

```

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.*

### 17.35.1 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](#).

### 17.36 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158

```

```

00166     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_

```

## 17.37 include/mtk\_quad\_1d.h File Reference

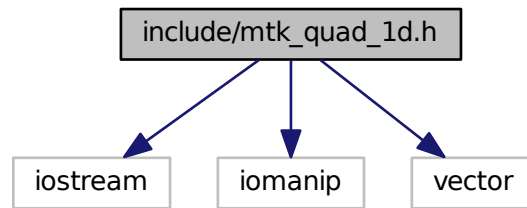
Includes the definition of the class Quad1D.

```

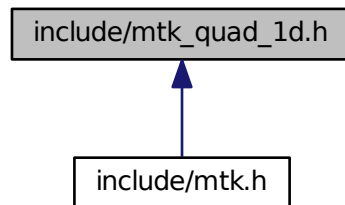
#include <iostream>
#include <iomanip>
#include <vector>

```

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.*

### 17.37.1 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](#).

## 17.38 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. 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:
00083     friend std::ostream& operator <<(std::ostream& stream, Quad1D &in);
00084
00085     Quad1D();
00086
00087     Quad1D(const Quad1D &quad);
00088
00089
00090
00091
00092
00093
00094
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_

```

## 17.39 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.*

### 17.39.1 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 mechanism.

Definition in file [mtk\\_roots.h](#).

## 17.40 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00032 list of conditions and the following disclaimer.
00033
00034 4. Redistributions in binary form must reproduce the above copyright notice,
00035 this list of conditions and the following disclaimer in the documentation and/or
00036 other materials provided with the distribution.
00037
00038 5. Usage of the binary form on proprietary applications shall require explicit
00039 prior written permission from the the copyright holders.
00040
00041 6. 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_

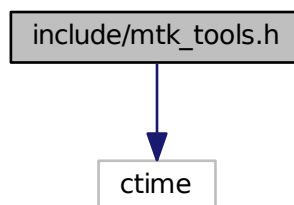
```

## 17.41 include/mtk\_tools.h File Reference

Tool manager class.

```
#include <ctime>
```

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.*

### 17.41.1 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.h`.

### 17.42 `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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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 namespace mtk {
00062
00072 class Tools {
00073 public:
00084     static void Prevent(const bool condition,
00085                        const char *fname,
00086                        int lineno,
00087                        const char *fxname);
00088
00094     static void BeginTestNo(const int &nn);
00095
00101     static void EndTestNo(const int &nn);
00102
00103 private:
00104     static int test_number_;
00105
00106     static clock_t begin_time_;
00107 };
00108 }
00109 #endif // End of: MTK_INCLUDE_TOOLS_H_

```

## 17.43 include/mtk\_uni\_stg\_grid\_1d.h File Reference

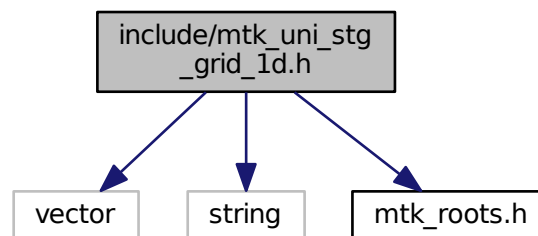
Definition of an 1D uniform staggered grid.

```

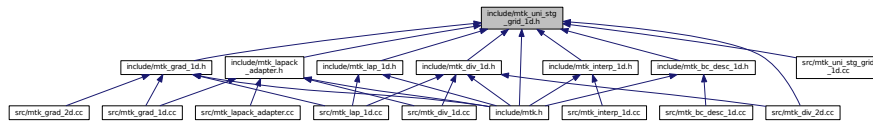
#include <vector>
#include <string>
#include "mtk_roots.h"

```

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.*

### 17.43.1 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`.

## 17.44 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
```

```

00035
00036 6. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078 public:
00080     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
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_

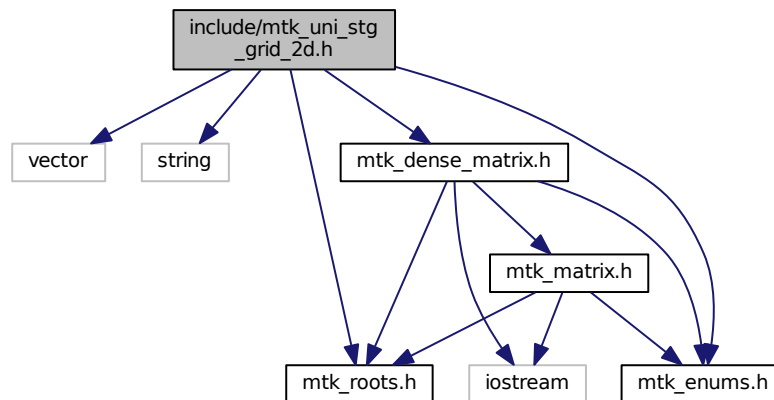
```

## 17.45 include/mtk\_uni\_stg\_grid\_2d.h File Reference

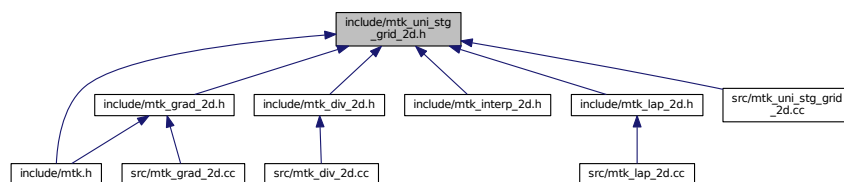
Definition of an 2D uniform staggered grid.

```
#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.*

### 17.45.1 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\\_2d.h](#).

## 17.46 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. 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 west_bndy_x() const;
00124
00130     Real east_bndy_x() const;
00131
00137     Real south_bndy_y() const;
00138
00144     Real north_bndy_y() const;
00145
00151     Real delta_x() const;
00152
00158     Real delta_y() const;
00159
00165     Real *discrete_domain_x();
00166
00172     Real *discrete_domain_y();
00173
00179     Real *discrete_field_u();
00180
00186     int num_cells_x() const;
00187
00193     int num_cells_y() const;
00194
00200     void BindScalarField(Real (*ScalarField)(Real xx, Real yy));
00201
00213     void BindVectorFieldPComponent(Real (*VectorField)(
00214 Real xx, Real yy));
00214
00226     void BindVectorFieldQComponent(Real (*VectorField)(
00227 Real xx, Real yy));
00227
00239     bool WriteToFile(std::string filename,
00240                     std::string space_name,
00241                     std::string field_name);
00242
00243 private:
00244     FieldNature nature_;
00245
00246     std::vector<Real> discrete_domain_x_;
00247     std::vector<Real> discrete_domain_y_;
00248     std::vector<Real> discrete_field_u_;
00249
00250     Real west_bndy_x_;
00251     Real east_bndy_x_;
00252     Real num_cells_x_;
00253     Real delta_x_;
00254
00255     Real south_bndy_y_;
00256     Real north_bndy_y_;
00257     Real num_cells_y_;
00258     Real delta_y_;
00259 };
00260 }
00261 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_

```

## 17.47 Makefile.inc File Reference



## 17.48 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... where is the MTK?
00008 # _____
00009
00010 BASE = /home/ejspeiro/Dropbox/MTK
00011
00012 # 2. The machine (platform) identifier and required 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-4.55/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-4.55/lib/libglpk.a
00056
00057 # If optimization is OFF, then provide the paths for:
00058
00059 BLAS_LIB = $(HOME)/Libraries/BLAS/libblas.a
00060 LAPACK_LIB = $(HOME)/Libraries/lapack-3.4.1/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 = colorgcc
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 -O2
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:
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!
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 DOCFEILNAME = doc_config.dxcf
00171 DOC         = $(BASE)/doc
00172 DOCFEIL     = $(BASE)/$(DOCFEILNAME)

```

## 17.49 README.md File Reference

### 17.50 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 discretization** 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 functional and more complete.
00018 -----
00019
00020 ## 2. Dependencies
00021
00022 This README assumes all of these dependencies are installed in the following
00023 folder:
00024 ```
00025 ```
00026 $(HOME)/Libraries/
00027 ```
00028
00029 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00030 routines for the internal computation on some of the layers. However, ATLAS
00031 requires both BLAS and LAPACK in order to create their optimized distributions.
00032 Therefore, the following dependencies tree arises:
00033
00034 ### For Linux:
00035
00036 1. LAPACK - Available from: http://www.netlib.org/lapack/
00037 1. BLAS - Available from: http://www.netlib.org/blas/
00038
00039 2. GLPK - Available from: https://www.gnu.org/software/glpk/
00040
00041 3. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00042 1. BLAS - Available from: http://www.netlib.org/blas/
00043 2. LAPACK - Available from: http://www.netlib.org/lapack/
00044
00045 4. (Optional) Valgrind - Available from: http://valgrind.org/
00046
00047 5. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00048
00049 ### For OS X:
00050
00051 1. GLPK - Available from: https://www.gnu.org/software/glpk/
00052 -----
00053
00054 ## 3. Installation
00055
00056 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00057

```

```

00058 The following steps are required the build and test the MTK. Please use the
00059 accompanying 'Makefile.inc' file, which should provide a solid template to
00060 start with. The following command provides help on the options for make:
00061 ```
00062 ```
00063 $ make help
00064 -----
00065 Makefile for the MTK.
00066
00067 Options are:
00068 - make: builds only the library and the examples.
00069 - all: builds the library, the examples and the documentation.
00070 - mtklib: builds the library, i.e. generates the archive files.
00071 - tests: generates the tests.
00072 - examples: generates the examples.
00073 - gendoc: generates the documentation for the library.
00074 - checkheaders: checks syntax of the header files.
00075
00076 - clean: cleans ALL the generated files.
00077 - cleanlib: cleans the generated archive and object files.
00078 - cleantests: cleans the generated tests executables.
00079 - cleanexamples: cleans the generated examples executables.
00080 -----
00081 ```
00082
00083 ### PART 2. BUILD THE LIBRARY.
00084
00085 ```
00086 $ make
00087 ```
00088
00089 If successful you'll read (before building the tests and examples):
00090
00091 ```
00092 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00093 ```
00094
00095 Examples and tests will also be built.
00096
00097 -----
00098 ## 4. Frequently Asked Questions
00099
00100 Q: Why haven't you guys implemented GBS to build the library?
00101 A: I'm on it as we speak! ;)
00102
00103 Q: When will the other flavors be ready?
00104 A: Soon! I'm working on getting help on developing those.
00105
00106 Q: Is there any main reference when it comes to the theory on Mimetic Methods?
00107 A: Yes! Check: http://www.csrc.sdsu.edu/mimetic-book
00108
00109 Q: Do I need to generate the documentation myself?
00110 A: You can if you want to... but if you DO NOT want to, just go to our website.
00111
00112 -----
00113 ## 5. Contact, Support, and Credits
00114
00115 The MTK is developed by researchers and adjuncts to the
00116 [Computational Science Research Center (CSRC)](http://www.csrc.sdsu.edu/)
00117 at [San Diego State University (SDSU)](http://www.sdsu.edu/).
00118
00119 Developers are members of:
00120
00121 1. Mimetic Numerical Methods Research and Development Group.
00122 2. Computational Geoscience Research and Development Group.
00123 3. Ocean Modeling Research and Development Group.
00124
00125 Currently the developers are:
00126
00127 - **Eduardo J. Sanchez, Ph.D. - esanchez@mail.sdsu.edu - @ejspeiro
00128 - Jose E. Castillo, Ph.D. - jcastillo@mail.sdsu.edu
00129 - Guillermo F. Miranda, Ph.D. - unigrav@hotmail.com
00130 - Christopher P. Paolini, Ph.D. - paolini@engineering.sdsu.edu
00131 - Angel Boada.
00132 - Johnny Corbino.
00133 - Raul Vargas-Navarro.
00134
00135 Finally, please feel free to contact me with suggestions or corrections:
00136
00137 **Eduardo J. Sanchez, Ph.D. - esanchez@mail.sdsu.edu - @ejspeiro
00138

```

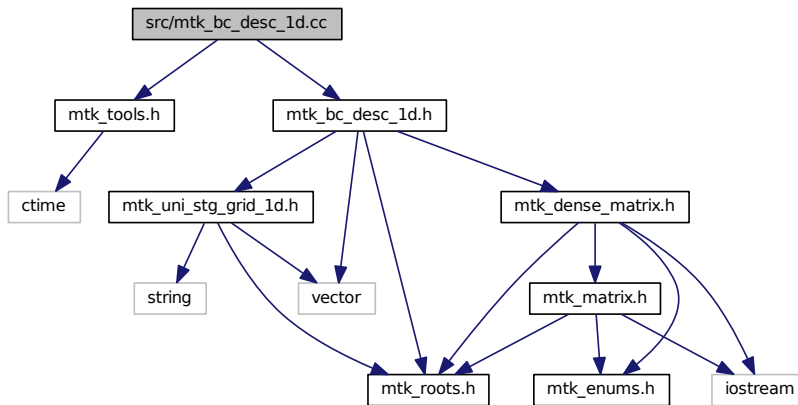
00139 Thanks and happy coding!

## 17.51 src/mtk\_bc\_desc\_1d.cc File Reference

```
#include "mtk_tools.h"
```

```
#include "mtk_bc_desc_1d.h"
```

Include dependency graph for mtk\_bc\_desc\_1d.cc:



## 17.52 mtk\_bc\_desc\_1d.cc

```

00001 #include "mtk_tools.h"
00002
00003 #include "mtk_bc_desc_1d.h"
00004
00005 void mtk::BCDesc1D::ImposeOnOperator(
00006     mtk::DenseMatrix &matrix,
00007     const std::vector<mtk::Real> &west,
00008     const std::vector<mtk::Real> &east) {
00009     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00010     mtk::Tools::Prevent(west.size() > (unsigned int) matrix.
00011         num_cols(),
00012         __FILE__, __LINE__, __func__);
00013     mtk::Tools::Prevent(east.size() > (unsigned int) matrix.
00014         num_cols(),
00015         __FILE__, __LINE__, __func__);
00016
00017     for (unsigned int ii = 0; ii < west.size(); ++ii) {
00018         matrix.SetValue(0, ii, west[ii]);
00019     }
00020
00021     for (unsigned int ii = 0; ii < east.size(); ++ii) {
00022         matrix.SetValue(matrix.num_rows() - 1,
00023             matrix.num_cols() - 1 - ii,
00024             east[ii]);
00025     }
00026 }
00027
00028 void mtk::BCDesc1D::ImposeOnGrid(mtk::UniStgGrid1D &grid,
00029     const mtk::Real &omega,
00030     const mtk::Real &epsilon) {

```

```

00033
00034  mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00035
00037
00038  grid.discrete_field_u()[0] = omega;
00039
00041
00042  grid.discrete_field_u()[grid.num_cells_x() + 2 - 1] = epsilon;
00043 }

```

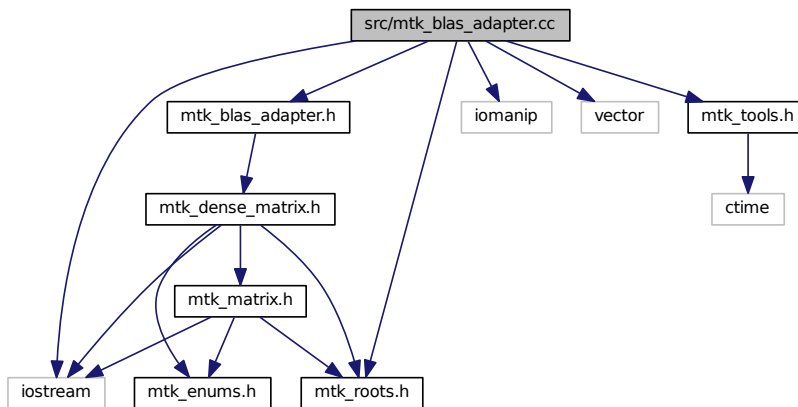
## 17.53 src/mtk\_blas\_adapter.cc File Reference

```

#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)

## 17.54 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. Documentation related to said modifications should be included.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions of source code must retain the above copyright notice, this
00039 list of conditions and the following disclaimer.
00040
00041 4. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 5. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders.
00047
00048 6. 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 dnm2_(int *n, double *x, int *incx);
00098 #else
00099
00112 float snrm2_(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(
    mtk::Real *computed,
                                mtk::Real *known,
                                int length) {
00359
00360
00361
00362  #if MTK_DEBUG_LEVEL > 0
00363  mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00364  mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00365  #endif
00366
00367  mtk::Real norm_2_computed(mtk::BLASAdapter::RealNRM2(known, length));
00368
00369  mtk::Real alpha{-mtk::kOne};
00370
00371  mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00372
00373  mtk::Real norm_2_difference(mtk::BLASAdapter::RealNRM2(computed,
length));
00374
00375  return norm_2_difference/norm_2_computed;
00376 }
00377
00378 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00379                                     mtk::DenseMatrix &aa,
00380                                     mtk::Real *xx,
00381                                     mtk::Real &beta,
00382                                     mtk::Real *yy) {
00383
00384  // Make sure input matrices are row-major ordered.
00385
00386  if (aa.matrix_properties().ordering() ==
mtk::COL_MAJOR) {
00387      aa.OrderRowMajor();
00388  }
00389
00390  char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00391
00392  int mm{aa.num_rows()}; // Rows of aa.
00393  int nn{aa.num_cols()}; // Columns of aa.
00394  int lda{(aa.matrix_properties()).ld()}; // Leading dimension.
00395  int incx{1}; // Increment of values in x.
00396  int incy{1}; // Increment of values in y.
00397
00398  std::swap(mm, nn);
00399  #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
00412  #if MTK_DEBUG_LEVEL > 0
00413  mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00414                      __FILE__, __LINE__, __func__);
00415  #endif
00416
00417  // Make sure input matrices are row-major ordered.
00418
00419  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 }

```

## 17.55 src/mtk\_dense\_matrix.cc File Reference

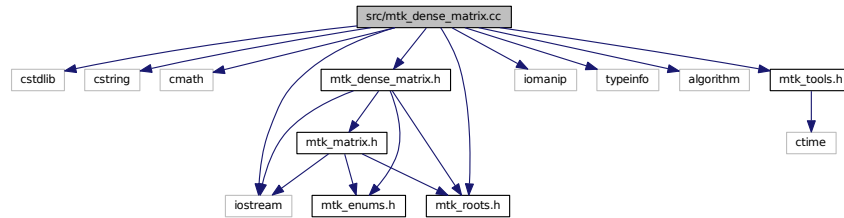
Implements a common dense matrix, using a 1D array.

```

#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#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)`

### 17.55.1 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

Definition in file [mtk\\_dense\\_matrix.cc](#).

## 17.56 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00028 list of conditions and the following disclaimer.
00029
00030 4. Redistributions in binary form must reproduce the above copyright notice,
00031 this list of conditions and the following disclaimer in the documentation and/or
  
```

```

00032 other materials provided with the distribution.
00033
00034 5. Usage of the binary form on proprietary applications shall require explicit
00035 prior written permission from the the copyright holders.
00036
00037 6. 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 <typeinfo>
00066
00067 #include <algorithm>
00068
00069 #include "mtk_roots.h"
00070 #include "mtk_dense_matrix.h"
00071 #include "mtk_tools.h"
00072
00073 namespace mtk {
00074
00075 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00076
00077     int mm{in.matrix_properties_.num_rows()}; // Auxiliary.
00078     int nn{in.matrix_properties_.num_cols()}; // Auxiliary.
00079
00080     if (in.matrix_properties_.ordering() ==
00081         mtk::COL_MAJOR) {
00082         std::swap(mm, nn);
00083     }
00084     for (auto ii = 0; ii < mm; ii++) {
00085         for (auto jj = 0; jj < nn; jj++) {
00086             mtk::Real value = in.data_[ii*nn + jj];
00087             stream << std::setw(13) << value;
00088         }
00089         stream << std::endl;
00090     }
00091     if (in.matrix_properties_.ordering() ==
00092         mtk::COL_MAJOR) {
00093         std::swap(mm, nn);
00094     }
00095     return stream;
00096 }
00097
00098 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
00099 mtk::DenseMatrix &in) {
00100
00101     if(this == &in) {
00102         return *this;
00103     }
00104
00105     matrix_properties_.set_storage(in.
00106 matrix_properties_.storage());
00107
00108     matrix_properties_.set_ordering(in.
00109 matrix_properties_.ordering());
00110
00111     auto aux = in.matrix_properties_.num_rows();

```

```

00108     matrix_properties_.set_num_rows(aux);
00109
00110     aux = in.matrix_properties().num_cols();
00111     matrix_properties_.set_num_cols(aux);
00112
00113     aux = in.matrix_properties().num_zero();
00114     matrix_properties_.set_num_zero(aux);
00115
00116     aux = in.matrix_properties().num_null();
00117     matrix_properties_.set_num_null(aux);
00118
00119     auto num_rows = matrix_properties_.num_rows();
00120     auto num_cols = matrix_properties_.num_cols();
00121
00122     delete [] data_;
00123
00124     try {
00125         data_ = new mtk::Real[num_rows*num_cols];
00126     } catch (std::bad_alloc &memory_allocation_exception) {
00127         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00128             std::endl;
00129         std::cerr << memory_allocation_exception.what() << std::endl;
00130     }
00131     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
num_cols);
00132
00133     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00134
00135     return *this;
00136 }
00137
00138 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00139
00140     matrix_properties_.set_storage(mtk::DENSE);
00141     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00142 }
00143
00144 mtk::DenseMatrix::DenseMatrix(const
mtk::DenseMatrix &in) {
00145
00146     matrix_properties_.set_storage(in.matrix_properties_.storage());
00147
00148     matrix_properties_.set_ordering(in.matrix_properties_.
ordering());
00149
00150     auto aux = in.matrix_properties_.num_rows();
00151     matrix_properties_.set_num_rows(aux);
00152
00153     aux = in.matrix_properties().num_cols();
00154     matrix_properties_.set_num_cols(aux);
00155
00156     aux = in.matrix_properties().num_zero();
00157     matrix_properties_.set_num_zero(aux);
00158
00159     aux = in.matrix_properties().num_null();
00160     matrix_properties_.set_num_null(aux);
00161
00162     auto num_rows = in.matrix_properties_.num_rows();
00163     auto num_cols = in.matrix_properties_.num_cols();
00164
00165     try {
00166         data_ = new mtk::Real[num_rows*num_cols];
00167     } catch (std::bad_alloc &memory_allocation_exception) {
00168         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00169             std::endl;
00170         std::cerr << memory_allocation_exception.what() << std::endl;
00171     }
00172     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00173
00174     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00175 }
00176
00177 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00178
00179     #if MTK_DEBUG_LEVEL > 0
00180     mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00181     mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00182     #endif
00183
00184     matrix_properties_.set_storage(mtk::DENSE);
00185     matrix_properties_.set_ordering(mtk::ROW_MAJOR);

```

```

00186 matrix_properties_.set_num_rows(num_rows);
00187 matrix_properties_.set_num_cols(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
00199 mtk::DenseMatrix::DenseMatrix(const int &rank,
00200                               const bool &padded,
00201                               const bool &transpose) {
00202
00203     #if MTK_DEBUG_LEVEL > 0
00204     mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00205     #endif
00206
00207     int aux{}; // Used to control the padding.
00208
00209     if (padded) {
00210         aux = 1;
00211     }
00212
00213     matrix_properties_.set_storage(mtk::DENSE);
00214     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00215     matrix_properties_.set_num_rows(aux + rank + aux);
00216     matrix_properties_.set_num_cols(rank);
00217
00218     try {
00219         data_ = new mtk::Real[matrix_properties_.num_values()];
00220     } catch (std::bad_alloc &memory_allocation_exception) {
00221         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00222         std::endl;
00223         std::cerr << memory_allocation_exception.what() << std::endl;
00224     }
00225     memset(data_,
00226            mtk::kZero,
00227            sizeof(data_[0])*(matrix_properties_.num_values()));
00228
00229     for (auto ii = 0; ii < matrix_properties_.num_rows(); ++ii) {
00230         for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00231             data_[ii*matrix_properties_.num_cols() + jj] =
00232                 (ii == jj + aux)? mtk::kOne: mtk::kZero;
00233         }
00234     }
00235 }
00236
00237 mtk::DenseMatrix::DenseMatrix(const mtk::Real *gen,
00238                               const int &gen_length,
00239                               const int &pro_length,
00240                               const bool &transpose) {
00241
00242     #if MTK_DEBUG_LEVEL > 0
00243     mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00244     mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00245     mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00246     #endif
00247
00248     matrix_properties_.set_storage(mtk::DENSE);
00249     matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00250     if (!transpose) {
00251         matrix_properties_.set_num_rows(gen_length);
00252         matrix_properties_.set_num_cols(pro_length);
00253     } else {
00254         matrix_properties_.set_num_rows(pro_length);
00255         matrix_properties_.set_num_cols(gen_length);
00256     }
00257
00258     int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00259     int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00260
00261     try {
00262         data_ = new mtk::Real[rr*cc];
00263     } catch (std::bad_alloc &memory_allocation_exception) {
00264         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00265         std::endl;
00266         std::cerr << memory_allocation_exception.what() << std::endl;

```

```

00267     }
00268     memset(data_, mtk::kZero, sizeof(data_[0])*rr*cc);
00269
00270     if (!transpose) {
00271         for (auto ii = 0; ii < rr; ii++) {
00272             for (auto jj = 0; jj < cc; jj++) {
00273                 data_[ii*cc + jj] = pow(gen[ii], (double) jj);
00274             }
00275         }
00276     } else {
00277         for (auto ii = 0; ii < rr; ii++) {
00278             for (auto jj = 0; jj < cc; jj++) {
00279                 data_[ii*cc + jj] = pow(gen[jj], (double) ii);
00280             }
00281         }
00282     }
00283 }
00284
00285 mtk::DenseMatrix::~DenseMatrix() {
00286     delete[] data_;
00287     data_ = nullptr;
00288 }
00289
00290
00291 mtk::Matrix mtk::DenseMatrix::matrix_properties() const {
00292     return matrix_properties_;
00293 }
00294
00295
00296 void mtk::DenseMatrix::SetOrdering(
    mtk::MatrixOrdering oo) {
00297
00298     #if MTK_DEBUG_LEVEL > 0
00299     mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
    mtk::COL_MAJOR),
00300         __FILE__, __LINE__, __func__);
00301     #endif
00302     matrix_properties_.set_ordering(oo);
00303 }
00304
00305
00306 int mtk::DenseMatrix::num_rows() const {
00307     return matrix_properties_.num_rows();
00308 }
00309
00310
00311 int mtk::DenseMatrix::num_cols() const {
00312     return matrix_properties_.num_cols();
00313 }
00314
00315
00316 mtk::Real* mtk::DenseMatrix::data() const {
00317     return data_;
00318 }
00319
00320
00321 mtk::Real mtk::DenseMatrix::GetValue(
    const int &rr,
    const int &cc) const {
00322
00323     #if MTK_DEBUG_LEVEL > 0
00324     mtk::Tools::Prevent(rr < 0, __FILE__, __LINE__, __func__);
00325     mtk::Tools::Prevent(cc < 0, __FILE__, __LINE__, __func__);
00326     #endif
00327     return data_[rr*matrix_properties_.num_cols() + cc];
00328 }
00329
00330
00331 void mtk::DenseMatrix::SetValue(
    const int &rr,
    const int &cc,
    const mtk::Real &val) {
00332
00333     #if MTK_DEBUG_LEVEL > 0
00334     mtk::Tools::Prevent(rr < 0, __FILE__, __LINE__, __func__);
00335     mtk::Tools::Prevent(cc < 0, __FILE__, __LINE__, __func__);
00336     #endif
00337     data_[rr*matrix_properties_.num_cols() + cc] = val;
00338 }
00339
00340
00341
00342
00343
00344
00345

```

```

00346 void mtk::DenseMatrix::Transpose() {
00347
00348
00349
00350     mtk::Real *data_transposed{}; // Buffer.
00351
00352     int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00353     int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00354
00355     try {
00356         data_transposed = new mtk::Real[rr*cc];
00357     } catch (std::bad_alloc &memory_allocation_exception) {
00358         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00359             std::endl;
00360         std::cerr << memory_allocation_exception.what() << std::endl;
00361     }
00362     memset(data_transposed,
00363         mtk::kZero,
00364         sizeof(data_transposed[0])*rr*cc);
00365
00366     // Assign the values to their transposed position.
00367     for (auto ii = 0; ii < rr; ++ii) {
00368         for (auto jj = 0; jj < cc; ++jj) {
00369             data_transposed[jj*rr + ii] = data_[ii*cc + jj];
00370         }
00371     }
00372
00373     // Swap pointers.
00374     auto tmp = data_; // Temporal holder.
00375     data_ = data_transposed;
00376     delete [] tmp;
00377     tmp = nullptr;
00378
00379     matrix_properties_.set_num_rows(cc);
00380     matrix_properties_.set_num_cols(rr);
00381 }
00382
00383 void mtk::DenseMatrix::OrderRowMajor() {
00384
00385     if (matrix_properties_.ordering() == mtk::COL_MAJOR) {
00386
00387
00388
00389         mtk::Real *data_transposed{}; // Buffer.
00390
00391         int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00392         int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00393
00394         try {
00395             data_transposed = new mtk::Real[rr*cc];
00396         } catch (std::bad_alloc &memory_allocation_exception) {
00397             std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00398                 std::endl;
00399             std::cerr << memory_allocation_exception.what() << std::endl;
00400         }
00401         memset(data_transposed,
00402             mtk::kZero,
00403             sizeof(data_transposed[0])*rr*cc);
00404
00405         // Assign the values to their transposed position.
00406         std::swap(rr, cc);
00407         for (auto ii = 0; ii < rr; ++ii) {
00408             for (auto jj = 0; jj < cc; ++jj) {
00409                 data_transposed[jj*rr + ii] = data_[ii*cc + jj];
00410             }
00411         }
00412         std::swap(rr, cc);
00413
00414         // Swap pointers.
00415         auto tmp = data_; // Temporal holder.
00416         data_ = data_transposed;
00417         delete [] tmp;
00418         tmp = nullptr;
00419
00420         matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00421     }
00422 }
00423
00424 void mtk::DenseMatrix::OrderColMajor() {
00425
00426     if (matrix_properties_.ordering() == ROW_MAJOR) {
00427
00428
00429

```



```

00430     mtk::Real *data_transposed{}; // Buffer.
00431
00432     int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00433     int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00434
00435     try {
00436         data_transposed = new mtk::Real[rr*cc];
00437     } catch (std::bad_alloc &memory_allocation_exception) {
00438         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00439             std::endl;
00440         std::cerr << memory_allocation_exception.what() << std::endl;
00441     }
00442     memset(data_transposed,
00443         mtk::kZero,
00444         sizeof(data_transposed[0])*rr*cc);
00445
00446     // Assign the values to their transposed position.
00447     for (auto ii = 0; ii < rr; ++ii) {
00448         for (auto jj = 0; jj < cc; ++jj) {
00449             data_transposed[jj*rr + ii] = data_[ii*cc + jj];
00450         }
00451     }
00452
00453     // Swap pointers.
00454     auto tmp = data_; // Temporal holder.
00455     data_ = data_transposed;
00456     delete [] tmp;
00457     tmp = nullptr;
00458
00459     matrix_properties_.set_ordering(mtk::COL_MAJOR);
00460 }
00461 }
00462
00463 mtk::DenseMatrix mtk::DenseMatrix::Kron(const
mtk::DenseMatrix &aa,
                                const mtk::DenseMatrix &bb) {
00464
00465     int row_offset{}; // Offset for rows.
00466     int col_offset{}; // Offset for rows.
00467
00468     mtk::Real aa_factor{}; // Used in computation.
00469
00470     // Auxiliary variables:
00471     auto aux1 = aa.matrix_properties_.num_rows()*bb.
matrix_properties_.num_rows();
00472     auto aux2 = aa.matrix_properties_.num_cols()*bb.
matrix_properties_.num_cols();
00473
00474     mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00475
00476     int kk_num_cols(output.matrix_properties_.num_cols()); // Aux.
00477
00478     auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00479     auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00480     auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00481     auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00482
00483     for (auto ii = 0; ii < mm; ++ii) {
00484         row_offset = ii*pp;
00485         for (auto jj = 0; jj < nn; ++jj) {
00486             col_offset = jj*qq;
00487             aa_factor = aa.data_[ii*nn + jj];
00488             for (auto ll = 0; ll < pp; ++ll) {
00489                 for (auto oo = 0; oo < qq; ++oo) {
00490                     auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00491                     output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00492                 }
00493             }
00494         }
00495     }
00496
00497     output.matrix_properties_.set_storage(mtk::DENSE);
00498     output.matrix_properties_.set_ordering(
mtk::ROW_MAJOR);
00499
00500     return output;
00501 }
00502 }
00503

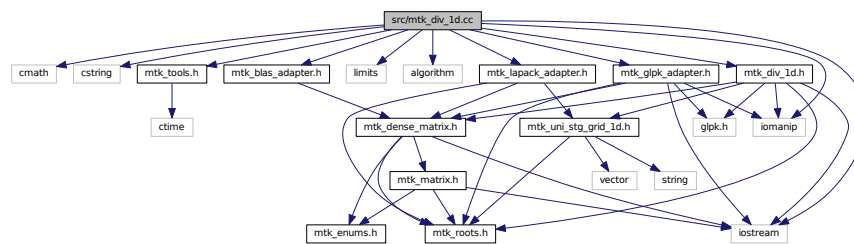
```

## 17.57 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)`

### 17.57.1 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](#).

## 17.58 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. 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_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00080
00082
00083     stream << "divergence_[0] = " << std::setw(9) << in.divergence_[0] <<
00084         std::endl;
00085
00087
00088     stream << "divergence_[1:" << in.order_accuracy_ << "]" = ";
00089     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00090         stream << std::setw(9) << in.divergence_[ii] << " ";
00091     }
00092     stream << std::endl;
00093

```

```

00094     if (in.order_accuracy_ > 2) {
00095
00096
00097         stream << "divergence_" << in.order_accuracy_ + 1 << ":" <<
00098             2*in.order_accuracy_ << "]" = ";
00099         for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00100             order_accuracy_; ++ii) {
00101             stream << std::setw(9) << in.divergence_[ii] << " ";
00102         }
00103         stream << std::endl;
00104
00105
00106         auto offset = (2*in.order_accuracy_ + 1);
00107         int mm{};
00108         for (auto ii = 0; ii < in.dim_null_; ++ii) {
00109             stream << "divergence_" << offset + mm << ":" <<
00110                 offset + mm + in.num_bndy_coeffs_ - 1 << "]" = ";
00111             for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {
00112                 auto value = in.divergence_[offset + mm];
00113                 stream << std::setw(9) << value << " ";
00114                 ++mm;
00115             }
00116             stream << std::endl;
00117         }
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
00196
00197     bool abort_construction = ComputeStencilInteriorGrid();
00198
00199     #if MTK_DEBUG_LEVEL > 0
00200     if (!abort_construction) {
00201         std::cerr << "Could NOT complete stage 1." << std::endl;
00202         std::cerr << "Exiting..." << std::endl;
00203         return false;
00204     }
00205     #endif
00206
00207     // At this point, we already have the values for the interior stencil stored
00208     // in the coeffs_interior_ array.
00209
00210     // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00211     // approximation at the boundary, thus it has no weights. For this case, the
00212     // dimension of the null-space of the Vandermonde matrices used to compute the
00213     // approximating coefficients at the boundary is 0. Ergo, we compute this
00214     // number first and then decide if we must compute anything at the boundary.
00215
00216     dim_null_ = order_accuracy_/2 - 1;
00217
00218     if (dim_null_ > 0) {
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
00226         // For this we will follow recommendations given in:
00227         //
00228         // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00229         //
00230         // We will compute the QR Factorization of the transpose, as in the
00231         // following (MATLAB) pseudo-code:
00232         //
00233         // [Q,R] = qr(V'); % Full QR as defined in
00234         // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00235         //
00236         // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00237         //
00238         // However, given the nature of the Vandermonde matrices we've just
00239         // computed, they all posses the same null-space. Therefore, we impose the
00240         // convention of computing the null-space of the first Vandermonde matrix
00241         // (west boundary).
00242
00243         abort_construction = ComputeRationalBasisNullSpace();
00244
00245         #if MTK_DEBUG_LEVEL > 0
00246         if (!abort_construction) {
00247             std::cerr << "Could NOT complete stage 2.1." << std::endl;
00248             std::cerr << "Exiting..." << std::endl;
00249             return false;
00250         }
00251         #endif
00252
00253         abort_construction = ComputePreliminaryApproximations();
00254
00255
00256
00257
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
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     abort_construction = ComputeStencilBoundaryGrid();
00282
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     return num_bndy_coeffs_;
00317 }
00318
00319 mtk::Real *mtk::Div1D::coeffs_interior() const {
00320     return coeffs_interior_;
00321 }
00322
00323 mtk::Real *mtk::Div1D::weights_crs() const {
00324     return weights_crs_;
00325 }
00326
00327 mtk::Real *mtk::Div1D::weights_cbs() const {
00328     return weights_cbs_;
00329 }
00330
00331 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00332     mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00333
00334     auto counter = 0;
00335     for (auto ii = 0; ii < dim_null_; ++ii) {
00336         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00337             xx.SetValue(ii,jj, divergence_[2*order_accuracy_ + 1 + counter]);
00338         }
00339     }
00340 }

```

```

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
00370
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
00385
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
00395
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
00415
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
00446 bool transpose{false};
00447
00448 mtk::DenseMatrix vander_matrix(pp,
00449                                 order_accuracy_,
00450                                 order_accuracy_,
00451                                 transpose);
00452
00453 #if MTK_DEBUG_LEVEL > 0
00454 std::cout << "vander_matrix =" << std::endl;
00455 std::cout << vander_matrix << std::endl;
00456 #endif
00457
00458 try {
00459     coeffs_interior_ = new mtk::Real[order_accuracy_];
00460 } catch (std::bad_alloc &memory_allocation_exception) {
00461     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00462         std::endl;
00463     std::cerr << memory_allocation_exception.what() << std::endl;
00464 }
00465 memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00466
00467 coeffs_interior_[1] = mtk::kOne;
00468
00469 #if MTK_DEBUG_LEVEL > 0
00470 std::cout << "oo =" << std::endl;
00471 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00472     std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00473 }
00474 std::cout << std::endl;
00475 #endif
00476
00477 int info(mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00478                                               coeffs_interior_));
00479
00480 #if MTK_DEBUG_LEVEL > 0
00481 if (!info) {
00482     std::cout << "System solved! Interior stencil attained!" << std::endl;
00483     std::cout << std::endl;
00484 }
00485 else {
00486     std::cerr << "Something wrong solving system! info =" << info << std::endl;
00487     std::cerr << "Exiting..." << std::endl;
00488     return false;
00489 }
00490 #endif
00491
00492 #if MTK_DEBUG_LEVEL > 0
00493 std::cout << "coeffs_interior_" << std::endl;
00494 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00495     std::cout << std::setw(12) << coeffs_interior_[ii];
00496 }
00497 std::cout << std::endl << std::endl;
00498 #endif
00499
00500 delete [] pp;
00501 pp = nullptr;
00502
00503 return true;
00504 }
00505
00506

```



```

00512 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
00513
00514     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00515
00516     try {
00517         gg = new mtk::Real[num_bndy_coeffs_];
00518     } catch (std::bad_alloc &memory_allocation_exception) {
00519         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00520             std::endl;
00521         std::cerr << memory_allocation_exception.what() << std::endl;
00522     }
00523     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00524
00525     #ifdef MTK_PRECISION_DOUBLE
00526     gg[0] = -1.0/2.0;
00527     #else
00528     gg[0] = -1.0f/2.0f;
00529     #endif
00530     for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00531         gg[ii] = gg[ii - 1] + mtk::kOne;
00532     }
00533
00534     #if MTK_DEBUG_LEVEL > 0
00535     std::cout << "gg =" << std::endl;
00536     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00537         std::cout << std::setw(12) << gg[ii];
00538     }
00539     std::cout << std::endl << std::endl;
00540     #endif
00541
00542     bool tran{true}; // Should I transpose the Vandermonde matrix.
00543
00544     mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00545
00546     #if MTK_DEBUG_LEVEL > 0
00547     std::cout << "vv_west_t =" << std::endl;
00548     std::cout << vv_west_t << std::endl;
00549     #endif
00550
00551     mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00552         (vv_west_t));
00553
00554     #if MTK_DEBUG_LEVEL > 0
00555     std::cout << "QQ^T =" << std::endl;
00556     std::cout << qq_t << std::endl;
00557     #endif
00558
00559     int KK_num_rows_{num_bndy_coeffs_};
00560     int KK_num_cols_{dim_null_};
00561
00562     mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00563
00564     for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00565         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00566             KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00567                 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00568         }
00569     }
00570
00571     #if MTK_DEBUG_LEVEL > 0
00572     std::cout << "KK =" << std::endl;
00573     std::cout << KK << std::endl;
00574     std::cout << "KK.num_rows() = " << KK.num_rows() << std::endl;
00575     std::cout << "KK.num_cols() = " << KK.num_cols() << std::endl;
00576     std::cout << std::endl;
00577     #endif
00578
00579     // Scale thus requesting that the last entries of the attained basis for the
00580     // null-space, adopt the pattern we require.
00581     // Essentially we will implement the following MATLAB pseudo-code:
00582     // scalers = KK(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00583     // SK = KK*scalers
00584     // where SK is the scaled null-space.
00585
00586     // In this point, we almost have all the data we need correctly allocated
00587     // in memory. We will create the matrix II_, and elements we wish to scale in

```

```

00597 // the KK array. Using the concept of the leading dimension, we could just
00598 // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00599 // GET how does it work. So I will just create a matrix with the content of
00600 // this array that we need, solve for the scalars and then scale the
00601 // whole KK:
00602
00603 // We will then create memory for that sub-matrix of KK (SUBK).
00604
00605 mtk::DenseMatrix SUBK(dim_null_,dim_null_);
00606
00607 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00608     for (auto jj = 0; jj < dim_null_; ++jj) {
00609         SUBK.data()[ (ii - (num_bndy_coeffs_ - dim_null_))*dim_null_ + jj] =
00610             KK.data()[ii*dim_null_ + jj];
00611     }
00612 }
00613
00614 #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     mtk::Real *gg{}; // Generator vector for the first approximation.
00691
00692     try {
00693         gg = new mtk::Real[num_bndy_coeffs_];
00694     } catch (std::bad_alloc &memory_allocation_exception) {
00695         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00696         std::endl;
00697         std::cerr << memory_allocation_exception.what() << std::endl;
00698     }
00699     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00700
00701     #ifdef MTK_PRECISION_DOUBLE
00702     gg[0] = -1.0/2.0;
00703     #else
00704     gg[0] = -1.0f/2.0f;
00705     #endif
00706     for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00707         gg[ii] = gg[ii - 1] + mtk::kOne;
00708     }
00709
00710     #if MTK_DEBUG_LEVEL > 0
00711     std::cout << "gg0 =" << std::endl;
00712     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00713         std::cout << std::setw(12) << gg[ii];
00714     }
00715     std::cout << std::endl << std::endl;
00716     #endif
00717
00718     // Allocate 2D array to store the collection of preliminary approximations.
00719     try {
00720         prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00721     } catch (std::bad_alloc &memory_allocation_exception) {
00722         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00723         std::endl;
00724         std::cerr << memory_allocation_exception.what() << std::endl;
00725     }
00726     memset(prem_apps_,
00727            mtk::kZero,
00728            sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00729
00730     for (auto ll = 0; ll < dim_null_; ++ll) {
00731
00732         // Re-check new generator vector for every iteration except for the first.
00733         #if MTK_DEBUG_LEVEL > 0
00734         if (ll > 0) {
00735             std::cout << "gg" << ll << " =" << std::endl;
00736             for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00737                 std::cout << std::setw(12) << gg[ii];
00738             }
00739             std::cout << std::endl << std::endl;
00740         }
00741         #endif
00742
00743         bool transpose{false};
00744
00745         mtk::DenseMatrix AA_(gg,
00746                               num_bndy_coeffs_, order_accuracy_ + 1,
00747                               transpose);
00748
00749         #if MTK_DEBUG_LEVEL > 0
00750         std::cout << "AA_" << ll << " =" << std::endl;
00751         std::cout << AA_ << std::endl;
00752         #endif
00753
00754         mtk::Real *ob{};
00755     }
00756 }

```

```

00763
00764     auto ob_ld = num_bndy_coeffs_;
00765
00766     try {
00767         ob = new mtk::Real[ob_ld];
00768     } catch (std::bad_alloc &memory_allocation_exception) {
00769         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00770             std::endl;
00771         std::cerr << memory_allocation_exception.what() << std::endl;
00772     }
00773     memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00774
00775     ob[1] = mtk::kOne;
00776
00777     #if MTK_DEBUG_LEVEL > 0
00778     std::cout << "ob = " << std::endl << std::endl;
00779     for (auto ii = 0; ii < ob_ld; ++ii) {
00780         std::cout << std::setw(12) << ob[ii] << std::endl;
00781     }
00782     std::cout << std::endl;
00783     #endif
00784
00785     // However, this is an under-determined system of equations. So we can not
00786     // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00787     // our LAPACKAdapter class.
00788
00789     int info_{
00790         mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00791             ob, ob_ld)};
00792
00793     #if MTK_DEBUG_LEVEL > 0
00794     if (!info_) {
00795         std::cout << "System successfully solved!" << std::endl << std::endl;
00796     } else {
00797         std::cerr << "Error solving system! info = " << info_ << std::endl;
00798     }
00799     #endif
00800
00801     #if MTK_DEBUG_LEVEL > 0
00802     std::cout << "ob =" << std::endl;
00803     for (auto ii = 0; ii < ob_ld; ++ii) {
00804         std::cout << std::setw(12) << ob[ii] << std::endl;
00805     }
00806     std::cout << std::endl;
00807     #endif
00808
00809     // This implies a DAXPY operation. However, we must construct the arguments
00810     // for this operation.
00811
00812     // Save them into the ob_bottom array:
00813
00814     Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00815
00816     try {
00817         ob_bottom = new mtk::Real[dim_null_];
00818     } catch (std::bad_alloc &memory_allocation_exception) {
00819         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00820             std::endl;
00821         std::cerr << memory_allocation_exception.what() << std::endl;
00822     }
00823     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00824
00825     for (auto ii = 0; ii < dim_null_; ++ii) {
00826         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00827     }
00828
00829     #if MTK_DEBUG_LEVEL > 0
00830     std::cout << "ob_bottom =" << std::endl;
00831     for (auto ii = 0; ii < dim_null_; ++ii) {
00832         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00833     }
00834     std::cout << std::endl;
00835     #endif
00836
00837     // We must computed an scaled ob, sob, using the scaled null-space in
00838     // rat_basis_null_space_.
00839     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00840     // or:
00841     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_ + 11] = 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 (11 = 0; 11 < dim_null; 11++);
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     // Assemble the pi matrix using:
00914     // 1. The collection of scaled preliminary approximations.
00915     // 2. The collection of coefficients approximating at the interior.
00916     // 3. The scaled basis for the null-space.
00917
00918     // 1.1. Process array of scaled preliminary approximations.
00919
00920     // These are queued in scaled_solutions. Each one of these, will be a column
00921     // of the pi matrix:
00922     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00923         for (auto jj = 0; jj < dim_null_; ++jj) {
00924             pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00925                 prem_apps_[ii*dim_null_ + jj];
00926         }
00927     }

```

```

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 + mm)*(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;

```

```

01012     std::cerr << memory_allocation_exception.what() << std::endl;
01013 }
01014 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01015
01016 int weights_ld{pi.num_cols() + 1};
01017
01018 // Preserve hh.
01019 std::copy(hh, hh + weights_ld, weights_cbs_);
01020
01021 pi.Transpose();
01022
01023 int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
pi, weights_cbs_, weights_ld)};
01024
01025 #if MTK_DEBUG_LEVEL > 0
01026 if (!info) {
01027     std::cout << "System successfully solved!" << std::endl << std::endl;
01028 } 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 <<
std::endl;
01047     std::cerr << memory_allocation_exception.what() << std::endl;
01048 }
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     mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_cbs_,
order_accuracy_)};
01065     mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01066
01067     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01068
01069     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01070         for (auto jj = 0; jj < dim_null_; ++jj) {
01071             phi.data()[ii*(order_accuracy_ + 1) + jj] = prem_apps_[ii*dim_null_ + jj];
01072         }
01073     }
01074
01075     int aux{}; // Auxiliary variable.
01076     for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01077         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01078             phi.data()[(ii + aux)*order_accuracy_ + jj] = coeffs_interior_[ii];
01079             ++aux;
01080         }
01081     }
01082
01083     for (auto jj = order_accuracy_ - 1; jj >= order_accuracy_ - dim_null_; jj--) {
01084         for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01085             phi.data()[ii*order_accuracy_ + jj] = mtk::kZero;
01086         }
01087     }
01088 }
01089
01090
01091
01092

```

```

01093     }
01094
01095     for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01096         for (auto ii = 0; ii < dim_null_; ++ii) {
01097             phi.data()[ (ii + order_accuracy_ - dim_null_ + jj*order_accuracy_) ] =
01098                 -prem_apps_[ (dim_null_ - ii - 1 + jj*dim_null_) ];
01099         }
01100     }
01101
01102     for (auto ii = 0; ii < order_accuracy_/2; ++ii) {
01103         for (auto jj = dim_null_ + 2; jj < order_accuracy_; ++jj) {
01104             auto swap = phi.data()[ ii*order_accuracy_+jj ];
01105             phi.data()[ ii*order_accuracy_ + jj ] =
01106                 phi.data()[ (order_accuracy_-ii)*order_accuracy_+jj ];
01107             phi.data()[ (order_accuracy_-ii)*order_accuracy_+jj ] = swap;
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
01116     mtk::Real *lamed{}; // Used to build big lambda.
01117
01118     try {
01119         lamed = new mtk::Real[dim_null_];
01120     } catch (std::bad_alloc &memory_allocation_exception) {
01121         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01122             std::endl;
01123         std::cerr << memory_allocation_exception.what() << std::endl;
01124     }
01125     memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01126
01127     for (auto ii = 0; ii < dim_null_; ++ii) {
01128         lamed[ii] = hh[ii + order_accuracy_ + 1];
01129     }
01130
01131     #if MTK_DEBUG_LEVEL > 0
01132     std::cout << "lamed =" << std::endl;
01133     for (auto ii = 0; ii < dim_null_; ++ii) {
01134         std::cout << std::setw(12) << lamed[ii] << std::endl;
01135     }
01136     std::cout << std::endl;
01137     #endif
01138
01139     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01140         mtk::Real temp = mtk::kZero;
01141         for (auto jj = 0; jj < dim_null_; ++jj) {
01142             temp = temp +
01143                 lamed[jj]*rat_basis_null_space_.data()[ jj*num_bndy_coeffs_ + ii ];
01144         }
01145         hh[ii] = hh[ii] - temp;
01146     }
01147
01148     #if MTK_DEBUG_LEVEL > 0
01149     std::cout << "big_lambda =" << std::endl;
01150     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01151         std::cout << std::setw(12) << hh[ii] << std::endl;
01152     }
01153     std::cout << std::endl;
01154     #endif
01155
01156     int copy_result{};
01157
01158     mtk::Real normerr_; // Norm of the error for the solution on each row.
01159
01160     for (auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01161         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01162             data(),
01163                 order_accuracy_ + 1,
01164                 order_accuracy_,
01165                 order_accuracy_,
01166                 hh,
01167                 weights_cbs_,
01168                 row_,
01169                 mimetic_threshold_,
01170                 copy_result);
01171         mtk::Real aux{normerr_/norm_};
01172     }

```



```

01175
01176     #if MTK_DEBUG_LEVEL>0
01177     std::cout << "Relative norm: " << aux << " " << std::endl;
01178     std::cout << std::endl;
01179     #endif
01180
01181     if (aux < minnorm_) {
01182         minnorm_ = aux;
01183         minrow_ = row_;
01184     }
01185 }
01186
01187     #if MTK_DEBUG_LEVEL > 0
01188     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01189     std::endl;
01190     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01191         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01192     }
01193     std::cout << std::endl;
01194     #endif
01195
01197     // After we know which row yields the smallest relative norm that row is
01198     // chosen to be the objective function and the result of the optimizer is
01199     // chosen to be the new weights_.
01200
01201     #if MTK_DEBUG_LEVEL > 0
01202     std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01203     minrow_ + 1 << std::endl;
01204     std::cout << std::endl;
01205     #endif
01206
01207     copy_result = 1;
01208     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01209 data(),
01210                                     order_accuracy_ + 1,
01211                                     order_accuracy_,
01212                                     order_accuracy_,
01213                                     hh,
01214                                     weights_cbs_,
01215                                     minrow_,
01216                                     mimetic_threshold_,
01217                                     copy_result);
01218     mtk::Real aux_{normerr_/norm_};
01219     #if MTK_DEBUG_LEVEL > 0
01220     std::cout << "Relative norm: " << aux_ << std::endl;
01221     std::cout << std::endl;
01222     #endif
01223
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     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01245
01246     try {
01247         lambda = new mtk::Real[dim_null_];
01248     } catch (std::bad_alloc &memory_allocation_exception) {
01249         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01250         std::endl;
01251         std::cerr << memory_allocation_exception.what() << std::endl;
01252     }
01253     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01254
01255

```

```

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
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_ + 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 divergence_[0] = order_accuracy_;
01364
01365 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01366     divergence_[ii + 1] = coeffs_interior_[ii];
01367 }
01368
01369 if (order_accuracy_ > 2) {
01370     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01371         divergence_[1 + order_accuracy_ + ii] = weights_cbs_[ii];
01372     }
01373 }
01374
01375 if (order_accuracy_ > 2) {
01376     auto offset = (2*order_accuracy_ + 1);
01377     int mm{};
01378     for (auto ii = 0; ii < dim_null_; ++ii) {
01379         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01380             divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01381             ++mm;
01382         }
01383     }
01384 }
01385
01386 #if MTK_DEBUG_LEVEL > 0
01387 std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01388 std::cout << std::endl;
01389 #endif
01390
01391 return true;
01401 }

```

## 17.59 src/mtk\_div\_2d.cc File Reference

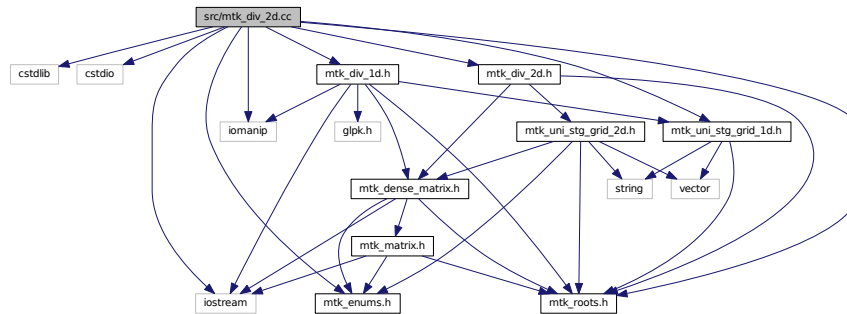
Implements the class Div2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"

```

Include dependency graph for `mtk_div_2d.cc`:



### 17.59.1 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](#).

## 17.60 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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 mtk::DenseMatrix mtk::Div2D::ConstructDiv2D(const
    mtk::UniStgGrid2D &grid,
                                int order_accuracy,
                                mtk::Real mimetic_threshold) {
00080
00081
00082
00083     int NumCellsX = grid.num_cells_x();
00084     int NumCellsY = grid.num_cells_y();
00085
00086     int mx = NumCellsX + 2; // Gx vertical dimension
00087     int nx = NumCellsX + 1; // Gx horizontal dimension
00088     int my = NumCellsY + 2; // Gy vertical dimension
00089     int ny = NumCellsY + 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     }
00098
00099     auto West = grid.west_bndy_x();
00100     auto East = grid.east_bndy_x();
00101     auto South = grid.south_bndy_y();
00102     auto North = grid.east_bndy_x();
00103
00104     mtk::UniStgGrid1D grid_x(West, East, NumCellsX);
00105     mtk::UniStgGrid1D grid_y(South, North, NumCellsY);
00106
00107     mtk::DenseMatrix Dx(div.ReturnAsDenseMatrix(grid_x));
00108     mtk::DenseMatrix Dy(div.ReturnAsDenseMatrix(grid_y));
00109
00110     bool padded{true};
00111     bool transpose{false};
00112
00113     mtk::DenseMatrix Ix(NumCellsX, padded, transpose);
00114     mtk::DenseMatrix Iy(NumCellsY, padded, transpose);
00115
00116     mtk::DenseMatrix Dxy(mtk::DenseMatrix::Kron(Iy, Dx));
00117     mtk::DenseMatrix Dyx(mtk::DenseMatrix::Kron(Dy, Ix));
00118
00119 #if MTK_DEBUG_LEVEL > 0
00120     std::cout << "Gx : " << mx << "by " << nx << std::endl;
00121     std::cout << "Transpose Iy : " << NumCellsY << " by " << ny << std::endl;
00122     std::cout << "Gy : " << my << "by " << ny << std::endl;
00123     std::cout << "Transpose Ix : " << NumCellsX << " by " << nx << std::endl;

```

```

00124     std::cout << "Kronecker dimensions Grad 2D" <<
00125     mx*NumCellsY + my*NumCellsX << " by " << nx*ny <<std::endl;
00126 #endif
00127
00128     mtk::DenseMatrix D2D(mx*my,nx*NumCellsY + ny*NumCellsX);
00129
00130     for (auto ii = 0; ii < mx*my; ii++) {
00131         for (auto jj = 0; jj < nx*NumCellsY; jj++) {
00132             D2D.SetValue(ii, jj, Dxy.GetValue(ii, jj));
00133         }
00134         for(auto kk=0; kk<ny*NumCellsX; kk++) {
00135             D2D.SetValue(ii, kk + nx*NumCellsY, Dyx.GetValue(ii, kk));
00136         }
00137     }
00138
00139     divergence_ = D2D;
00140
00141     return divergence_;
00142 }
00143
00144 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() {
00145     return divergence_;
00146 }
00147 }

```

## 17.61 src/mtk\_glpk\_adapter.cc File Reference

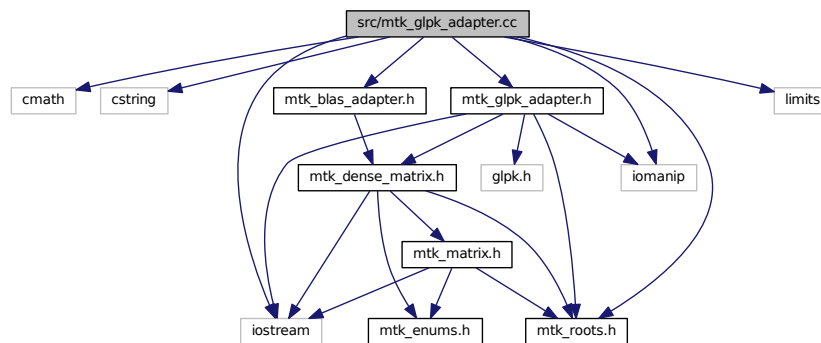
Adapter class for the GLPK API.

```

#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:



### 17.61.1 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](#).

## 17.62 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00034 list of conditions and the following disclaimer.
00035
00036 4. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 5. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders.
00042
00043 6. 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,
    int nrows,
    int ncols,
    int kk,
    mtk::Real *hh,
    mtk::Real *qq,
    int robjective,
    mtk::Real mimetic_threshold,
    int copy) {
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
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
00243
00244     for (ii = 1; ii <= lp_ncols; ++ii) {
00245         glp_set_obj_coef (lp, ii, objective[ii]);
00246     }
00247
00249
00250     for (ii = 1; ii <= lp_ncols; ++ii) {
00251         glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00252     }
00253
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
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 }

```

## 17.63 src/mtk\_grad\_1d.cc File Reference

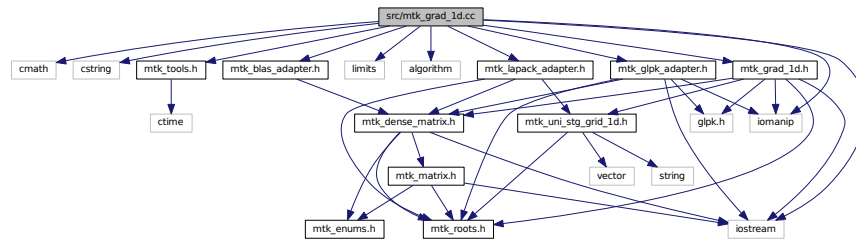
Implements the class Grad1D.

```

#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)`

### 17.63.1 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`.

## 17.64 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. 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_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &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     }
00118     else {
00119         stream << std::setw(9) << in.gradient_[offset + 0] << ' ';
00120         stream << std::setw(9) << in.gradient_[offset + 1] << ' ';
00121         stream << std::setw(9) << in.gradient_[offset + 2] << ' ';
00122     }
00123     stream << std::endl;
00124
00125     return stream;
00126 }
00127 }
00128
00129 mtk::Grad1D::Grad1D():
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::Grad1D::Grad1D(const Grad1D &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::GradID::~GradID() {
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::GradID::ConstructGradID(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_approxs_ = 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
00300 // Once we have the following three collections of data:
00301 // (a) the coefficients for the interior,
00302 // (b) the coefficients for the boundary (if it applies),
00303 // (c) and the weights (if it applies),
00304 // we will store everything in the output array:
00305
00306 abort_construction = AssembleOperator();
00307
00308 #if MTK_DEBUG_LEVEL > 0
00309 if (!abort_construction) {
00310     std::cerr << "Could NOT complete stage 3." << std::endl;
00311     std::cerr << "Exiting..." << std::endl;
00312     return false;
00313 }
00314 #endif
00315
00316 return true;
00317 }
00318
00319 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
00435     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00436     #endif
00437
00438     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00439
00440     int gg_num_rows = nn + 1;
00441     int gg_num_cols = nn + 2;
00442     int elements_per_row = num_bndy_coeffs_;
00443     int num_extra_rows = order_accuracy_/2;
00444
00445     // Output matrix featuring sizes for gradient operators.
00446     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
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
00489     return out;
00490 }
00491

```

```

00492 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00493
00495
00496     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00497
00498     try {
00499         pp = new mtk::Real[order_accuracy_];
00500     } catch (std::bad_alloc &memory_allocation_exception) {
00501         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00502             std::endl;
00503         std::cerr << memory_allocation_exception.what() << std::endl;
00504     }
00505     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00506
00507     #ifdef MTK_PRECISION_DOUBLE
00508     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00509     #else
00510     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00511     #endif
00512
00513     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00514         pp[ii] = pp[ii - 1] + mtk::kOne;
00515     }
00516
00517     #if MTK_DEBUG_LEVEL > 0
00518     std::cout << "pp =" << std::endl;
00519     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00520         std::cout << std::setw(12) << pp[ii];
00521     }
00522     std::cout << std::endl << std::endl;
00523     #endif
00524
00526
00527     bool transpose{false};
00528
00529     mtk::DenseMatrix vander_matrix(pp, order_accuracy_, order_accuracy_, transpose);
00530
00531     #if MTK_DEBUG_LEVEL > 0
00532     std::cout << "vander_matrix = " << std::endl;
00533     std::cout << vander_matrix << std::endl << std::endl;
00534     #endif
00535
00537
00538     try {
00539         coeffs_interior_ = new mtk::Real[order_accuracy_];
00540     } catch (std::bad_alloc &memory_allocation_exception) {
00541         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00542             std::endl;
00543         std::cerr << memory_allocation_exception.what() << std::endl;
00544     }
00545     memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00546
00547     coeffs_interior_[1] = mtk::kOne;
00548
00549     #if MTK_DEBUG_LEVEL > 0
00550     std::cout << "oo =" << std::endl;
00551     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00552         std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00553     }
00554     std::cout << std::endl;
00555     #endif
00556
00558
00559     int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00560                                                 coeffs_interior_)};
00561
00562     #if MTK_DEBUG_LEVEL > 0
00563     if (!info) {
00564         std::cout << "System solved! Interior stencil attained!" << std::endl;
00565         std::cout << std::endl;
00566     }
00567     else {
00568         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00569         std::cerr << "Exiting..." << std::endl;
00570         return false;
00571     }
00572     #endif
00573
00574     #if MTK_DEBUG_LEVEL > 0
00575     std::cout << "coeffs_interior_ =" << std::endl;
00576     for (auto ii = 0; ii < order_accuracy_; ++ii) {

```

```

00577     std::cout << std::setw(12) << coeffs_interior_[ii];
00578 }
00579 std::cout << std::endl << std::endl;
00580 #endif
00581
00582 delete [] pp;
00583 pp = nullptr;
00584
00585 return true;
00586 }
00587
00588 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00589
00590
00591     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00592
00593     try {
00594         gg = new mtk::Real[num_bndy_coeffs_];
00595     } catch (std::bad_alloc &memory_allocation_exception) {
00596         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00597             std::endl;
00598         std::cerr << memory_allocation_exception.what() << std::endl;
00599     }
00600     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00601
00602     #ifdef MTK_PRECISION_DOUBLE
00603     gg[1] = 1.0/2.0;
00604     #else
00605     gg[1] = 1.0f/2.0f;
00606     #endif
00607     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00608         gg[ii] = gg[ii - 1] + mtk::kOne;
00609     }
00610
00611     #if MTK_DEBUG_LEVEL > 0
00612     std::cout << "gg =" << std::endl;
00613     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00614         std::cout << std::setw(12) << gg[ii];
00615     }
00616     std::cout << std::endl << std::endl;
00617     #endif
00618
00619     bool tran{true}; // Should I transpose the Vandermonde matrix.
00620
00621     mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00622
00623     #if MTK_DEBUG_LEVEL > 0
00624     std::cout << "aa_west_t =" << std::endl;
00625     std::cout << aa_west_t << std::endl;
00626     #endif
00627
00628     mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00629         (aa_west_t));
00630
00631     #if MTK_DEBUG_LEVEL > 0
00632     std::cout << "qq_t =" << std::endl;
00633     std::cout << qq_t << std::endl;
00634     #endif
00635
00636     int kk_num_rows{num_bndy_coeffs_};
00637     int kk_num_cols{dim_null_};
00638
00639     mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00640
00641     // In the case of the gradient, even though we must solve for a null-space
00642     // of dimension 2, we must only extract ONE basis for the kernel.
00643     // We perform this extraction here:
00644
00645     int aux_{kk_num_rows - kk_num_cols};
00646     for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00647         aux_--;
00648         for (auto jj = 0; jj < kk_num_rows; jj++) {
00649             kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =
00650                 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00651         }
00652     }
00653
00654     #if MTK_DEBUG_LEVEL > 0

```

```

00661     std::cout << "kk =" << std::endl;
00662     std::cout << kk << std::endl;
00663     std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00664     std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00665     std::cout << std::endl;
00666     #endif
00667
00668
00669
00670     // Scale thus requesting that the last entries of the attained basis for the
00671     // null-space, adopt the pattern we require.
00672     // Essentially we will implement the following MATLAB pseudo-code:
00673     // scalers = kk(num_bndy_approx - (dim_null - 1):num_bndy_approx,:)\B
00674     // SK = kk*scalers
00675     // where SK is the scaled null-space.
00676
00677     // In this point, we almost have all the data we need correctly allocated
00678     // in memory. We will create the matrix iden_, and elements we wish to scale in
00679     // the kk array. Using the concept of the leading dimension, we could just
00680     // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00681     // GET how does it work. So I will just create a matrix with the content of
00682     // this array that we need, solve for the scalers and then scale the
00683     // whole kk:
00684
00685     // We will then create memory for that sub-matrix of kk (subk).
00686
00687     mtk::DenseMatrix subk(dim_null_, dim_null_);
00688
00689     auto zz = 0;
00690     for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00691         for (auto jj = 0; jj < dim_null_; jj++) {
00692             subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00693         }
00694         zz++;
00695     }
00696
00697     #if MTK_DEBUG_LEVEL > 0
00698     std::cout << "subk =" << std::endl;
00699     std::cout << subk << std::endl;
00700     #endif
00701
00702     subk.Transpose();
00703
00704     #if MTK_DEBUG_LEVEL > 0
00705     std::cout << "subk_t =" << std::endl;
00706     std::cout << subk << std::endl;
00707     #endif
00708
00709     bool padded{false};
00710     tran = false;
00711
00712     mtk::DenseMatrix iden(dim_null_, padded, tran);
00713
00714     #if MTK_DEBUG_LEVEL > 0
00715     std::cout << "iden =" << std::endl;
00716     std::cout << iden << std::endl;
00717     #endif
00718
00719     // Solve the system to compute the scalers.
00720     // An example of the system to solve, for k = 8, is:
00721     //
00722     // subk*scalers = iden or
00723     //
00724     // | 0.386018 -0.0339244 -0.129478 |           | 1 0 0 |
00725     // | -0.119774 0.0199423 0.0558632 |*scalers = | 0 1 0 |
00726     // | 0.0155708 -0.00349546 -0.00853182 |         | 0 0 1 |
00727     //
00728     // Notice this is a nrhs = 3 system.
00729     // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00730     // will be stored in the created identity matrix.
00731     // Let us first transpose subk (because of LAPACK):
00732
00733     int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00734
00735     #if MTK_DEBUG_LEVEL > 0
00736     if (!info) {
00737         std::cout << "System successfully solved!" <<
00738             std::endl;
00739     } else {
00740         std::cerr << "Something went wrong solving system! info = " << info <<
00741             std::endl;
00742         std::cerr << "Exiting..." << std::endl;

```

```

00743     return false;
00744 }
00745 std::cout << std::endl;
00746 #endif
00747
00748 #if MTK_DEBUG_LEVEL > 0
00749 std::cout << "Computed scalars:" << std::endl;
00750 std::cout << iden << std::endl;
00751 #endif
00752
00753 // Multiply the two matrices to attain a scaled basis for null-space.
00754
00755 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00756
00757 #if MTK_DEBUG_LEVEL > 0
00758 std::cout << "Rational basis for the null-space:" << std::endl;
00759 std::cout << rat_basis_null_space_ << std::endl;
00760 #endif
00761
00762 // At this point, we have a rational basis for the null-space, with the
00763 // pattern we need! :)
00764
00765 delete [] gg;
00766 gg = nullptr;
00767
00768 return true;
00769 }
00770
00771 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00772
00773     mtk::Real *gg{}; // Generator vector for the first approximation.
00774
00775     try {
00776         gg = new mtk::Real[num_bndy_coeffs_];
00777     } catch (std::bad_alloc &memory_allocation_exception) {
00778         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00779             std::endl;
00780         std::cerr << memory_allocation_exception.what() << std::endl;
00781     }
00782     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00783
00784 #ifdef MTK_PRECISION_DOUBLE
00785     gg[1] = 1.0/2.0;
00786 #else
00787     gg[1] = 1.0f/2.0f;
00788 #endif
00789 #endif
00790     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00791         gg[ii] = gg[ii - 1] + mtk::kOne;
00792     }
00793
00794 #if MTK_DEBUG_LEVEL > 0
00795 std::cout << "gg0 =" << std::endl;
00796 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00797     std::cout << std::setw(12) << gg[ii];
00798 }
00799 std::cout << std::endl << std::endl;
00800 #endif
00801
00802 // Allocate 2D array to store the collection of preliminary approximations.
00803
00804     try {
00805         prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00806     } catch (std::bad_alloc &memory_allocation_exception) {
00807         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00808             std::endl;
00809         std::cerr << memory_allocation_exception.what() << std::endl;
00810     }
00811     memset(prem_apps_,
00812         mtk::kZero,
00813         sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00814
00815     for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00816
00817         // Re-check new generator vector for every iteration except for the first.
00818         #if MTK_DEBUG_LEVEL > 0
00819         if (ll > 0) {
00820             std::cout << "gg" << ll << " =" << std::endl;
00821             for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00822                 std::cout << std::setw(12) << gg[ii];
00823             }
00824         }
00825

```

```

00826     std::cout << std::endl << std::endl;
00827 }
00828 #endif
00829
00831
00832 bool transpose{false};
00833
00834 mtk::DenseMatrix aa(gg,
00835                     num_bndy_coeffs_, order_accuracy_ + 1,
00836                     transpose);
00837
00838 #if MTK_DEBUG_LEVEL > 0
00839 std::cout << "aa_" << ll << " = " << std::endl;
00840 std::cout << aa << std::endl;
00841 #endif
00842
00844
00845 mtk::Real *ob{};
00846
00847 auto ob_ld = num_bndy_coeffs_;
00848
00849 try {
00850     ob = new mtk::Real[ob_ld];
00851 } catch (std::bad_alloc &memory_allocation_exception) {
00852     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00853         std::endl;
00854     std::cerr << memory_allocation_exception.what() << std::endl;
00855 }
00856 memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00857
00858 ob[1] = mtk::kOne;
00859
00860 #if MTK_DEBUG_LEVEL > 0
00861 std::cout << "ob = " << std::endl << std::endl;
00862 for (auto ii = 0; ii < ob_ld; ++ii) {
00863     std::cout << std::setw(12) << ob[ii] << std::endl;
00864 }
00865 std::cout << std::endl;
00866 #endif
00867
00869
00870 // However, this is an under-determined system of equations. So we can not
00871 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00872 // our LAPACKAdapter class.
00873
00874 int info_{
00875     mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
, ob_ld)};
00876
00877 #if MTK_DEBUG_LEVEL > 0
00878 if (!info_) {
00879     std::cout << "System successfully solved!" << std::endl << std::endl;
00880 } else {
00881     std::cerr << "Error solving system! info = " << info_ << std::endl;
00882 }
00883 #endif
00884
00885 #if MTK_DEBUG_LEVEL > 0
00886 std::cout << "ob =" << std::endl;
00887 for (auto ii = 0; ii < ob_ld; ++ii) {
00888     std::cout << std::setw(12) << ob[ii] << std::endl;
00889 }
00890 std::cout << std::endl;
00891 #endif
00892
00894
00895 // This implies a DAXPY operation. However, we must construct the arguments
00896 // for this operation.
00897
00899 // Save them into the ob_bottom array:
00900
00901 Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00902
00903 try {
00904     ob_bottom = new mtk::Real[dim_null_];
00905 } catch (std::bad_alloc &memory_allocation_exception) {
00906     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00907         std::endl;
00908     std::cerr << memory_allocation_exception.what() << std::endl;
00909 }
00910 memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);

```

```

00911
00912     for (auto ii = 0; ii < dim_null_; ++ii) {
00913         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00914     }
00915
00916     #if MTK_DEBUG_LEVEL > 0
00917     std::cout << "ob_bottom =" << std::endl;
00918     for (auto ii = 0; ii < dim_null_; ++ii) {
00919         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00920     }
00921     std::cout << std::endl;
00922     #endif
00923
00924     // We must computed an scaled ob, sob, using the scaled null-space in
00925     // rat_basis_null_space_.
00926     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00927     // or:                      ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00928     // thus:                      Y =      a*A      *x      +      b*Y (DAXPY).
00929
00930     #if MTK_DEBUG_LEVEL > 0
00931     std::cout << "Rational basis for the null-space:" << std::endl;
00932     std::cout << rat_basis_null_space_ << std::endl;
00933     #endif
00934
00935     mtk::Real alpha{-mtk::kOne};
00936     mtk::Real beta{mtk::kOne};
00937
00938     mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00939                                     ob_bottom, beta, ob);
00940
00941     #if MTK_DEBUG_LEVEL > 0
00942     std::cout << "scaled ob:" << std::endl;
00943     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00944         std::cout << std::setw(12) << ob[ii] << std::endl;
00945     }
00946     std::cout << std::endl;
00947     #endif
00948
00949     // We save the recently scaled solution, into an array containing these.
00950     // We can NOT start building the pi matrix, simply because I want that part
00951     // to be separated since its construction depends on the algorithm we want
00952     // to implement.
00953
00954     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00955         prem_apps_[ii*num_bndy_approxs_ + 11] = ob[ii];
00956     }
00957
00958     // After the first iteration, simply shift the entries of the last
00959     // generator vector used:
00960     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00961         gg[ii]--;
00962     }
00963
00964     // Garbage collection for this loop:
00965     delete[] ob;
00966     ob = nullptr;
00967
00968     delete[] ob_bottom;
00969     ob_bottom = nullptr;
00970
00971     } // End of: for (11 = 0; 11 < dim_null; 11++);
00972
00973     #if MTK_DEBUG_LEVEL > 0
00974     std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00975     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00976         for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
00977             std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
00978         }
00979         std::cout << std::endl;
00980     }
00981     std::cout << std::endl;
00982     #endif
00983
00984     delete[] gg;
00985     gg = nullptr;
00986
00987     return true;
00988 }
00989
00990 bool mtk::Grad1D::ComputeWeights() {
00991
00992

```

```

00993 // Matrix to compute the weights as in the CRSA.
00994 mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00995
00997
00998 // Assemble the pi matrix using:
00999 // 1. The collection of scaled preliminary approximations.
01000 // 2. The collection of coefficients approximating at the interior.
01001 // 3. The scaled basis for the null-space.
01002
01003 // 1.1. Process array of scaled preliminary approximations.
01004
01005 // These are queued in scaled_solutions. Each one of these, will be a column
01006 // of the pi matrix:
01007 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01008     for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01009         pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01010             prem_apps_[ii*num_bndy_approxs_ + jj];
01011     }
01012 }
01013
01014 // 1.2. Add columns from known stencil approximating at the interior.
01015
01016 // However, these must be padded by zeros, according to their position in the
01017 // final pi matrix:
01018 auto mm = 1;
01019 for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
01020     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01021         auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
01022             (order_accuracy_/2 + 1)) + jj;
01023         pi.data()[de] = coeffs_interior_[ii];
01024     }
01025     ++mm;
01026 }
01027
01028 rat_basis_null_space_.OrderColMajor();
01029
01030 #if MTK_DEBUG_LEVEL > 0
01031 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01032 std::cout << rat_basis_null_space_ << std::endl;
01033 #endif
01034
01035 // 1.3. Add final set of columns: rational basis for null-space.
01036
01037 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01038     jj < num_bndy_coeffs_ - 1; ++jj) {
01039     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01040         auto og =
01041             (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01042         auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01043         pi.data()[de] = rat_basis_null_space_.data()[og];
01044     }
01045 }
01046
01047 #if MTK_DEBUG_LEVEL > 0
01048 std::cout << "coeffs_interior_ =" << std::endl;
01049 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01050     std::cout << std::setw(12) << coeffs_interior_[ii];
01051 }
01052 std::cout << std::endl << std::endl;
01053 #endif
01054
01055 #if MTK_DEBUG_LEVEL > 0
01056 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01057 std::cout << pi << std::endl;
01058 #endif
01059
01060 // This imposes the mimetic condition.
01061
01062 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01063
01064 try {
01065     hh = new mtk::Real[num_bndy_coeffs_];
01066 } catch (std::bad_alloc &memory_allocation_exception) {
01067     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01068         std::endl;
01069     std::cerr << memory_allocation_exception.what() << std::endl;
01070 }
01071
01072 memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01073
01074 hh[0] = -mtk::kOne;

```



```

01076     for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01077         auto aux_xx = mtk::kZero;
01078         for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01079             aux_xx += coeffs_interior_[jj];
01080         }
01081         hh[ii] = -mtk::kOne*aux_xx;
01082     }
01083
01084 // That is, we construct a system, to solve for the weights.
01085
01086 // Once again we face the challenge of solving with LAPACK. However, for the
01087 // CRSA, this matrix PI is over-determined, since it has more rows than
01088 // unknowns. However, according to the theory, the solution to this system is
01089 // unique. We will use dgels_.
01090
01091 try {
01092     weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01093 } catch (std::bad_alloc &memory_allocation_exception) {
01094     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01095         std::endl;
01096     std::cerr << memory_allocation_exception.what() << std::endl;
01097 }
01098 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01099
01100 int weights_ld{pi.num_cols() + 1};
01101
01102 // Preserve hh.
01103 std::copy(hh, hh + weights_ld, weights_cbs_);
01104
01105 pi.Transpose();
01106
01107 int info{
01108     mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01109         weights_cbs_, weights_ld)
01110 };
01111
01112 #if MTK_DEBUG_LEVEL > 0
01113 if (!info) {
01114     std::cout << "System successfully solved!" << std::endl << std::endl;
01115 } else {
01116     std::cerr << "Error solving system! info = " << info << std::endl;
01117 }
01118 #endif
01119
01120 #if MTK_DEBUG_LEVEL > 0
01121 std::cout << "hh = " << std::endl;
01122 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01123     std::cout << std::setw(11) << hh[ii] << std::endl;
01124 }
01125 std::cout << std::endl;
01126 #endif
01127
01128 // Preserve the original weights for research.
01129
01130 try {
01131     weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01132 } catch (std::bad_alloc &memory_allocation_exception) {
01133     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01134         std::endl;
01135     std::cerr << memory_allocation_exception.what() << std::endl;
01136 }
01137 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01138
01139 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01140
01141 #if MTK_DEBUG_LEVEL > 0
01142 std::cout << "weights_CRSA + lambda = " << std::endl;
01143 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01144     std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01145 }
01146 std::cout << std::endl;
01147 #endif
01148
01149 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01150     int minrow{std::numeric_limits<int>::infinity()};
01151     mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01152         order_accuracy_)};

```

```

01158     mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01159
01161
01162     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01163
01164     // 6.1. Insert preliminary approximations to first set of columns.
01165
01166     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01167         for (auto jj = 0; jj < num_bndy_approx_ + 1; ++jj) {
01168             phi.data()[ii*(order_accuracy_ + 1) + jj] =
01169                 prem_apps[ii*num_bndy_approx_ + jj];
01170         }
01171     }
01172
01173     // 6.2. Skip a column and negate preliminary approximations.
01174
01175     for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01176         for (auto ii = 1; ii < num_bndy_approx_ + 1; ++ii) {
01177             auto de = (ii + order_accuracy_ - num_bndy_approx_ + jj*order_accuracy_);
01178             auto og = (num_bndy_approx_ - ii + (jj)*num_bndy_approx_);
01179             phi.data()[de] = -pre_apps[og];
01180         }
01181     }
01182
01183     // 6.3. Flip negative columns up-down.
01184
01185     for (auto ii = 0; ii < order_accuracy_/2; ++ii) {
01186         for (auto jj = num_bndy_approx_ + 1; jj < order_accuracy_; ++jj) {
01187             auto aux = phi.data()[ii*order_accuracy_ + jj];
01188             phi.data()[ii*order_accuracy_ + jj] =
01189                 phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01190             phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01191         }
01192     }
01193
01194     // 6.4. Insert stencil.
01195
01196     auto mm = 0;
01197     for (auto jj = num_bndy_approx_; jj < num_bndy_approx_ + 1; ++jj) {
01198         for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01199             if (ii == 0) {
01200                 phi.data()[jj] = 0.0;
01201             } else {
01202                 phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior[ii - 1];
01203             }
01204             mm++;
01205         }
01206     }
01207
01208     #if MTK_DEBUG_LEVEL > 0
01209     std::cout << "phi =" << std::endl;
01210     std::cout << phi << std::endl;
01211     #endif
01212
01213
01214
01215     mtk::Real *lamed{}; // Used to build big lambda.
01216
01217     try {
01218         lamed = new mtk::Real[num_bndy_approx_ - 1];
01219     } catch (std::bad_alloc &memory_allocation_exception) {
01220         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01221             std::endl;
01222         std::cerr << memory_allocation_exception.what() << std::endl;
01223     }
01224     memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approx_ - 1));
01225
01226     for (auto ii = 0; ii < num_bndy_approx_ - 1; ++ii) {
01227         lamed[ii] = hh[ii + order_accuracy_ + 1];
01228     }
01229
01230     #if MTK_DEBUG_LEVEL > 0
01231     std::cout << "lamed =" << std::endl;
01232     for (auto ii = 0; ii < num_bndy_approx_ - 1; ++ii) {
01233         std::cout << std::setw(12) << lamed[ii] << std::endl;
01234     }
01235     std::cout << std::endl;
01236     #endif
01237
01238     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01239         mtk::Real temp = mtk::kZero;
01240         for (auto jj = 0; jj < num_bndy_approx_ - 1; ++jj) {

```

```

01241         temp = temp +
01242             lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01243     }
01244     hh[ii] = hh[ii] - temp;
01245 }
01246
01247 #if MTK_DEBUG_LEVEL > 0
01248 std::cout << "big_lambda =" << std::endl;
01249 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01250     std::cout << std::setw(12) << hh[ii] << std::endl;
01251 }
01252 std::cout << std::endl;
01253 #endif
01254
01255 int copy_result{}; // Should I replace the solution... not for now.
01256
01257 mtk::Real normerr_; // Norm of the error for the solution on each row.
01258
01259 for (auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01260     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01261 data(),
01262                                     order_accuracy_ + 1,
01263                                     order_accuracy_,
01264                                     order_accuracy_,
01265                                     hh,
01266                                     weights_cbs_,
01267                                     row_,
01268                                     mimetic_threshold_,
01269                                     copy_result);
01270
01271     mtk::Real aux{normerr_/norm};
01272
01273     #if MTK_DEBUG_LEVEL>0
01274     std::cout << "Relative norm: " << aux << " " << std::endl;
01275     std::cout << std::endl;
01276     #endif
01277
01278     if (aux < minnorm) {
01279         minnorm = aux;
01280         minrow_ = row_;
01281     }
01282 }
01283
01284 #if MTK_DEBUG_LEVEL > 0
01285 std::cout << "weights_CBSA + lambda (after brute force search):" <<
01286     std::endl;
01287 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01288     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01289 }
01290 std::cout << std::endl;
01291 #endif
01292
01293 // After we know which row yields the smallest relative norm that row is
01294 // chosen to be the objective function and the result of the optimizer is
01295 // chosen to be the new weights_.
01296
01297 #if MTK_DEBUG_LEVEL > 0
01298 std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01299     minrow_ + 1 << std::endl;
01300 std::cout << std::endl;
01301 #endif
01302
01303 copy_result = 1;
01304 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01305 data(),
01306                                     order_accuracy_ + 1,
01307                                     order_accuracy_,
01308                                     order_accuracy_,
01309                                     hh,
01310                                     weights_cbs_,
01311                                     minrow_,
01312                                     mimetic_threshold_,
01313                                     copy_result);
01314
01315     mtk::Real aux_{normerr_/norm};
01316     #if MTK_DEBUG_LEVEL > 0
01317     std::cout << "Relative norm: " << aux_ << std::endl;
01318     std::cout << std::endl;
01319     #endif
01320
01321     delete [] lamed;

```

```

01322     lamed = nullptr;
01323 }
01324
01325 delete [] hh;
01326 hh = nullptr;
01327
01328 return true;
01329 }
01330
01331 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01332
01333     #if MTK_DEBUG_LEVEL > 0
01334     std::cout << "weights_* + lambda =" << std::endl;
01335     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01336         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01337     }
01338     std::cout << std::endl;
01339     #endif
01340
01341     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01342
01343     try {
01344         lambda = new mtk::Real[dim_null_];
01345     } catch (std::bad_alloc &memory_allocation_exception) {
01346         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01347             std::endl;
01348         std::cerr << memory_allocation_exception.what() << std::endl;
01349     }
01350     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01351
01352     for (auto ii = 0; ii < dim_null_; ++ii) {
01353         lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01354     }
01355
01356     #if MTK_DEBUG_LEVEL > 0
01357     std::cout << "lambda =" << std::endl;
01358     for (auto ii = 0; ii < dim_null_; ++ii) {
01359         std::cout << std::setw(12) << lambda[ii] << std::endl;
01360     }
01361     std::cout << std::endl;
01362     #endif
01363
01364     mtk::Real *alpha{}; // Collection of alpha values.
01365
01366     try {
01367         alpha = new mtk::Real[dim_null_];
01368     } catch (std::bad_alloc &memory_allocation_exception) {
01369         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01370             std::endl;
01371         std::cerr << memory_allocation_exception.what() << std::endl;
01372     }
01373     memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01374
01375     for (auto ii = 0; ii < dim_null_; ++ii) {
01376         alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01377     }
01378
01379     #if MTK_DEBUG_LEVEL > 0
01380     std::cout << "alpha =" << std::endl;
01381     for (auto ii = 0; ii < dim_null_; ++ii) {
01382         std::cout << std::setw(12) << alpha[ii] << std::endl;
01383     }
01384     std::cout << std::endl;
01385     #endif
01386
01387     try {
01388         mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01389     } catch (std::bad_alloc &memory_allocation_exception) {
01390         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01391             std::endl;
01392         std::cerr << memory_allocation_exception.what() << std::endl;
01393     }
01394     memset(mim_bndy_,
01395         mtk::kZero,
01396         sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01397
01398     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01399         for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {

```

```

01406     mim_bndy_[ii*num_bndy_approxs_ + jj] =
01407     prem_apps_[ii*num_bndy_approxs_ + jj] +
01408     alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01409 }
01410 }
01411
01412 for(auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01413     mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01414     prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01415 }
01416
01417 #if MTK_DEBUG_LEVEL > 0
01418 std::cout << "Collection of mimetic approximations:" << std::endl;
01419 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01420     for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01421         std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01422     }
01423     std::cout << std::endl;
01424 }
01425 std::cout << std::endl;
01426 #endif
01427
01428 delete[] lambda;
01429 lambda = nullptr;
01430
01431 delete[] alpha;
01432 alpha = nullptr;
01433
01434 return true;
01435 }
01436
01437 bool mtk::Grad1D::AssembleOperator(void) {
01438
01439     // The output array will have this form:
01440     // 1. The first entry of the array will contain the used order kk.
01441     // 2. The second entry of the array will contain the collection of
01442     // approximating coefficients for the interior of the grid.
01443     // 3. The third entry will contain a collection of weights.
01444     // 4. The next dim_null - 1 entries will contain the collections of
01445     // approximating coefficients for the west boundary of the grid.
01446
01447     gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01448     num_bndy_approxs_*num_bndy_coeffs_;
01449
01450     #if MTK_DEBUG_LEVEL > 0
01451     std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01452     #endif
01453
01454     try {
01455         gradient_ = new mtk::Real[gradient_length_];
01456     } catch (std::bad_alloc &memory_allocation_exception) {
01457         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01458         std::endl;
01459         std::cerr << memory_allocation_exception.what() << std::endl;
01460     }
01461     memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01462
01463
01464     gradient_[0] = order_accuracy_;
01465
01466
01467     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01470         gradient_[ii + 1] = coeffs_interior_[ii];
01471     }
01472
01473
01474     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01477         gradient_[(order_accuracy_ + 1) + ii] = weights_cbs_[ii];
01478     }
01479
01480     int offset{2*order_accuracy_ + 1};
01481
01482     int aux {}; // Auxiliary variable.
01483
01484     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01487         for (auto ii = 0; ii < num_bndy_approxs_ ; ii++) {
01488             for (auto jj = 0; jj < num_bndy_coeffs_ ; jj++) {
01489                 gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01490                 aux++;
01491             }
01492         }

```

```

01493     }
01494   } else {
01495     gradient_[offset + 0] = prem_apps_[0];
01496     gradient_[offset + 1] = prem_apps_[1];
01497     gradient_[offset + 2] = prem_apps_[2];
01498   }
01499
01500   #if MTK_DEBUG_LEVEL > 0
01501   std::cout << "1D " << order_accuracy_ << "--order grad built!" << std::endl;
01502   std::cout << std::endl;
01503   #endif
01504
01505   return true;
01506 }

```

## 17.65 src/mtk\_grad\_2d.cc File Reference

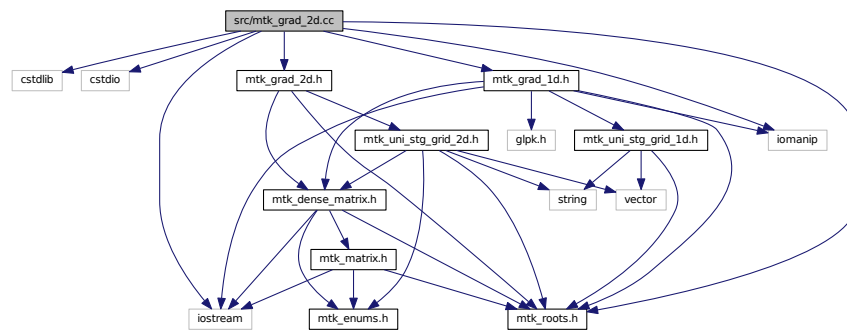
Implements the class Grad2D.

```

#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:



### 17.65.1 Detailed Description

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

#### Author

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

Definition in file [mtk\\_grad\\_2d.cc](#).

## 17.66 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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 mtk::DenseMatrix mtk::Grad2D::ConstructGrad2D(const
00078     mtk::UniStgGrid2D &grid,
00079     int order_accuracy,
00080     mtk::Real mimetic_threshold) {
00081     int NumCellsX = grid.num_cells_x();
00082     int NumCellsY = grid.num_cells_y();
00083
00084     int mx = NumCellsX + 1; // Gx vertical dimension
00085     int nx = NumCellsX + 2; // Gx horizontal dimension
00086     int my = NumCellsY + 1; // Gy vertical dimension

```

```

00087     int ny = NumCellsY + 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     }
00096
00097     auto West = grid.west_bndy_x();
00098     auto East = grid.east_bndy_x();
00099     auto South = grid.south_bndy_y();
00100     auto North = grid.east_bndy_x();
00101
00102     mtk::UniStgGrid1D grid_x(West, East, NumCellsX);
00103     mtk::UniStgGrid1D grid_y(South, North, NumCellsY);
00104
00105     mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00106     mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00107
00108     bool padded{true};
00109     bool transpose{true};
00110
00111     mtk::DenseMatrix TIx(NumCellsX, padded, transpose);
00112     mtk::DenseMatrix TIy(NumCellsY, padded, transpose);
00113
00114     mtk::DenseMatrix Gxy(mtk::DenseMatrix::Kron(TIy, Gx));
00115     mtk::DenseMatrix Gyx(mtk::DenseMatrix::Kron(Gy, TIx));
00116
00117     #if MTK_DEBUG_LEVEL > 0
00118     std::cout << "Gx : " << mx << "by " << nx << std::endl;
00119     std::cout << "Transpose Iy : " << NumCellsY << " by " << ny << std::endl;
00120     std::cout << "Gy : " << my << "by " << ny << std::endl;
00121     std::cout << "Transpose Ix : " << NumCellsX << " by " << nx << std::endl;
00122     std::cout << "Kronecker dimensions Grad 2D" <<
00123     mx*NumCellsY + my*NumCellsX << " by " << nx*ny << std::endl;
00124     #endif
00125
00126     mtk::DenseMatrix G2D(mx*NumCellsY + my*NumCellsX, nx*ny);
00127
00128     for(auto ii = 0; ii < nx*ny; ii++) {
00129         for(auto jj = 0; jj < mx*NumCellsY; jj++) {
00130             G2D.SetValue(jj,ii, Gxy.GetValue(jj,ii));
00131         }
00132         for(auto kk = 0; kk < my*NumCellsX; kk++) {
00133             G2D.SetValue(kk + mx*NumCellsY, ii, Gyx.GetValue(kk,ii));
00134         }
00135     }
00136
00137     gradient_ = G2D;
00138
00139     return gradient_;
00140 }
00141
00142 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix() {
00143
00144     return gradient_;
00145 }

```

## 17.67 src/mtk\_interp\_1d.cc File Reference

Includes the implementation of the class Interp1D.

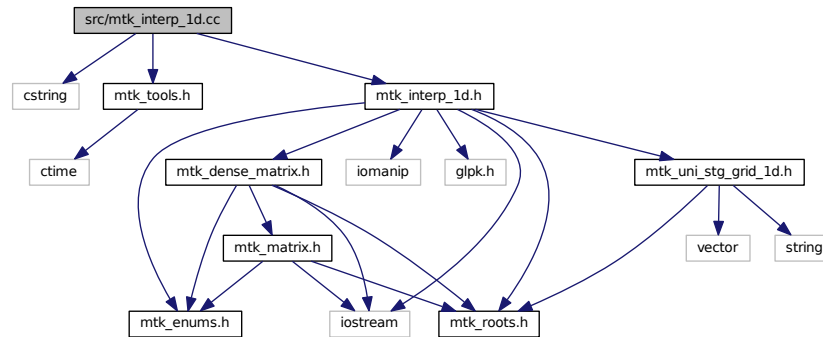
```

#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)`

### 17.67.1 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](#).

## 17.68 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #include <cstring>
00059
00060 #include "mtk_tools.h"
00061
00062 #include "mtk_interp_1d.h"
00063
00064 namespace mtk {
00065
00066 std::ostream& operator <<(std::ostream &stream, mtk::Interp1D &in) {
00067
00068     stream << "coeffs_interior_[1:" << in.order_accuracy_ << "] = ";
00069     for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00070         stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00071     }
00072     stream << std::endl;
00073
00074     return stream;
00075 }
00076
00077 mtk::Interp1D::Interp1D():
00078     dir_interp_(mtk::SCALAR_TO_VECTOR),
00079     order_accuracy_(mtk::kDefaultOrderAccuracy),
00080     coeffs_interior_(nullptr) {}
00081
00082 mtk::Interp1D::Interp1D(const Interp1D &interp):
00083     dir_interp_(interp.dir_interp_),
00084     order_accuracy_(interp.order_accuracy_),
00085     coeffs_interior_(interp.coeffs_interior_) {}
00086
00087 mtk::Interp1D::~Interp1D() {
00088     delete[] coeffs_interior_;
00089     coeffs_interior_ = nullptr;
00090 }
00091
00092 bool mtk::Interp1D::ConstructInterp1D(int order_accuracy,
00093     mtk::DirInterp dir) {
00094
00095     #if MTK_DEBUG_LEVEL > 0
00096     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00097     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00098     mtk::Tools::Prevent(dir < mtk::SCALAR_TO_VECTOR &&
00099         dir > mtk::VECTOR_TO_SCALAR,
00100         __FILE__, __LINE__, __func__);
00101
00102     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00103     #endif
00104 }

```

```

00107
00108     order_accuracy_ = order_accuracy;
00109
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
    UniStgGrid1D &grid) {
00136
00137     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00138
00139     #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     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00156
00157
00158
00159     out.SetValue(0, 0, mtk::kOne);
00160
00161
00162
00163     for (auto ii = 1; ii < gg_num_rows - 1; ++ii) {
00164         for (auto jj = ii; jj < order_accuracy_ + ii; ++jj) {
00165             out.SetValue(ii, jj, mtk::kOne/order_accuracy_);
00166         }
00167     }
00168
00169
00170
00171     out.SetValue(gg_num_rows - 1, gg_num_cols - 1, mtk::kOne);
00172
00173     return out;
00174 }

```

## 17.69 src/mtk\_lap\_1d.cc File Reference

Includes the implementation of the class Lap1D.



```

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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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::Lapl1D &in) {
00074
00075
00076
00077     stream << "laplacian_[0] = " << in.laplacian_[0] << std::endl << std::endl;
00078
00079
00080
00081     stream << "laplacian_[1:" << 2*in.order_accuracy_ - 1 << "]" = " <<
00082         std::endl << std::endl;
00083     for (auto ii = 1; ii <= (2*in.order_accuracy_ - 1); ++ii) {
00084         stream << std::setw(13) << in.laplacian_[ii] << " ";
00085     }
00086     stream << std::endl << std::endl;
00087
00088
00089
00090     auto offset = 1 + (2*in.order_accuracy_ - 1);
00091
00092     stream << "laplacian_[ " << offset << ":" << offset +
00093         (in.order_accuracy_ - 1)*(2*in.order_accuracy_) - 1 << "]" = " <<
00094         std::endl << std::endl;
00095
00096     for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00097         for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00098             stream << std::setw(13) <<

```

```

00099         in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj];
00100     }
00101     stream << std::endl;
00102 }
00103
00104 return stream;
00105 }
00106 }
00107
00108 mtk::LaplD::LaplD():
00109     order_accuracy_(mtk::kDefaultOrderAccuracy),
00110     laplacian_length_(),
00111     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00112
00113 mtk::LaplD::~~LaplD() {
00114     delete [] laplacian_;
00115     laplacian_ = nullptr;
00116 }
00117
00118
00119 bool mtk::LaplD::ConstructLaplD(int order_accuracy,
00120                                 mtk::Real mimetic_threshold) {
00121
00122     #if MTK_DEBUG_LEVEL > 0
00123     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00124     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00125     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00126                         __FILE__, __LINE__, __func__);
00127
00128     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00129         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00130     }
00131
00132     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00133     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00134     #endif
00135
00136     order_accuracy_ = order_accuracy;
00137     mimetic_threshold_ = mimetic_threshold;
00138
00139     mtk::Grad1D grad; // Mimetic gradient.
00140
00141     bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00142
00143     if (!info) {
00144         std::cerr << "Mimetic grad could not be built." << std::endl;
00145         return false;
00146     }
00147
00148     mtk::Div1D div; // Mimetic divergence.
00149
00150     info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00151
00152     if (!info) {
00153         std::cerr << "Mimetic div could not be built." << std::endl;
00154         return false;
00155     }
00156
00157     // Since these are mimetic operator, we must multiply the matrices arising
00158     // from both the divergence and the Laplacian, in order to get the
00159     // approximating coefficients for the Laplacian operator.
00160
00161     // However, we must choose a grid that implied a step size of 1, so to get
00162     // the approximating coefficients, without being affected from the
00163     // normalization with respect to the grid.
00164
00165     // Also, the grid must be of the minimum size to support the requested order
00166     // of accuracy. We must please the divergence.
00167
00168     mtk::UniStgGrid1D aux(mtk::kZero,
00169                          (mtk::Real) 3*order_accuracy_ - 1,
00170                          3*order_accuracy_ - 1);
00171
00172     #if MTK_DEBUG_LEVEL > 0
00173     std::cout << "aux = " << std::endl;
00174     std::cout << aux << std::endl;
00175     std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00176     std::cout << std::endl;
00177
00178     #endif
00179 }

```

```

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
00203     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
00215
00216     // The output array will have this form:
00217     // 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
00242     laplacian_[0] = order_accuracy_;
00243
00246
00247     for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00248         laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00249     }
00250
00252
00253     auto offset = 1 + (2*order_accuracy_ - 1);
00254
00255     for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00256         for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00257             laplacian_[offset + ii*(2*order_accuracy_) + jj] =
00258                 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
00280  auto offset = (1 + 2*order_accuracy_ - 1);
00281
00282  for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00283      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00284          lap.SetValue(1 + ii,
00285                      jj,
00286                      idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00287      }
00288  }
00289
00291
00292  offset = 1 + (order_accuracy_ - 1);
00293
00294  int kk{1};
00295  for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00296      int mm{1};
00297      for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00298          lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00299          mm = mm + 1;
00300      }
00301      kk = kk + 1;
00302  }
00303
00305
00306  offset = (1 + 2*order_accuracy_ - 1);
00307
00308  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
00328
00329  return lap;
00330 }
00331
00332 mtk::Real* mtk::LaplD::Data(const UniStgGrid1D &grid) {
00333
00334     mtk::DenseMatrix tmp;
00335
00336     tmp = ReturnAsDenseMatrix(grid);
00337
00338     return tmp.data();
00339 }

```

## 17.71 src/mtk\_lap\_2d.cc File Reference

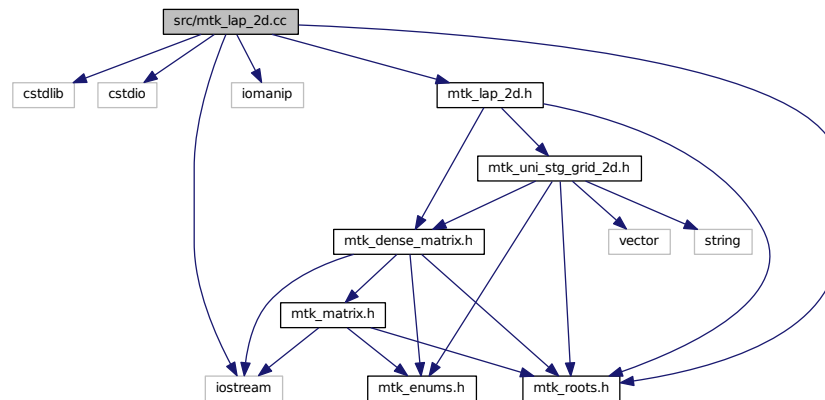
```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_lap_2d.h"

```



Include dependency graph for mtk\_lap\_2d.cc:



## 17.72 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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. 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 <stdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_lap_2d.h"
00065
00066 mtk::Lap2D::Lap2D():
00067     order_accuracy_(),
00068     mimetic_threshold_() {}
00069
00070 mtk::Lap2D::Lap2D(const Lap2D &lap):
00071     order_accuracy_(lap.order_accuracy_),
00072     mimetic_threshold_(lap.mimetic_threshold_) {}
00073
00074 mtk::Lap2D::~~Lap2D() {}
00075
00076 mtk::DenseMatrix mtk::Lap2D::ConstructLap2D(const
    mtk::UniStgGrid2D &grid,
00077                                         int order_accuracy,
00078                                         mtk::Real mimetic_threshold) {
00079
00080     return laplacian_;
00081 }

```

## 17.73 src/mtk\_lapack\_adapter.cc File Reference

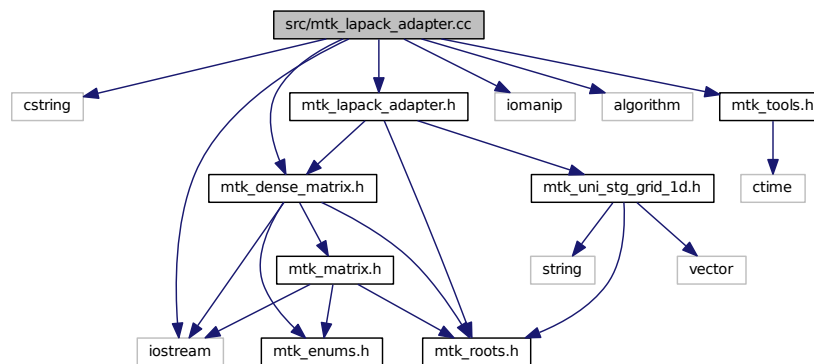
Adapter class for the LAPACK API.

```

#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.*

### 17.73.1 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.cc](#).

## 17.74 mtk\_lapack\_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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00034 list of conditions and the following disclaimer.
00035
00036 4. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 5. Usage of the binary form on proprietary applications shall require explicit
```

```

00041 prior written permission from the the copyright holders.
00042
00043 6. 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 <cstring>
00066
00067 #include <iostream>
00068 #include <iomanip>
00069
00070 #include <algorithm>
00071
00072 #include "mtk_tools.h"
00073 #include "mtk_dense_matrix.h"
00074 #include "mtk_lapack_adapter.h"
00075
00076 namespace mtk {
00077
00078 extern "C" {
00079
00080 #ifdef MTK_PRECISION_DOUBLE
00081
00100 void dgesv_(int* n,
00101             int* nrhs,
00102             Real* a,
00103             int* lda,
00104             int* ipiv,
00105             Real* b,
00106             int* ldb,
00107             int* info);
00108 #else
00109
00128 void sgesv_(int* n,
00129             int* nrhs,
00130             Real* a,
00131             int* lda,
00132             int* ipiv,
00133             Real* b,
00134             int* ldb,
00135             int* info);
00136 #endif
00137
00138 #ifdef MTK_PRECISION_DOUBLE
00139
00182 void dgels_(char* trans,
00183             int* m,
00184             int* n,
00185             int* nrhs,
00186             Real* a,
00187             int* lda,
00188             Real* b,
00189             int* ldb,
00190             Real* work,
00191             int* lwork,
00192             int* info);
00193 #else
00194
00237 void sgels_(char* trans,
00238             int* m,
00239             int* n,
00240             int* nrhs,
00241             Real* a,

```

```

00242         int* lda,
00243         Real* b,
00244         int* ldb,
00245         Real* work,
00246         int* lwork,
00247         int* info);
00248 #endif
00249
00250 #ifdef MTK_PRECISION_DOUBLE
00251
00280 void dgeqrf_(int *m,
00281              int *n,
00282              Real *a,
00283              int *lda,
00284              Real *tau,
00285              Real *work,
00286              int *lwork,
00287              int *info);
00288 #else
00289
00318 void sgeqrf_(int *m,
00319              int *n,
00320              Real *a,
00321              int *lda,
00322              Real *tau,
00323              Real *work,
00324              int *lwork,
00325              int *info);
00326 #endif
00327
00328 #ifdef MTK_PRECISION_DOUBLE
00329
00363 void dormqr_(char *side,
00364              char *trans,
00365              int *m,
00366              int *n,
00367              int *k,
00368              Real *a,
00369              int *lda,
00370              Real *tau,
00371              Real *c,
00372              int *ldc,
00373              Real *work,
00374              int *lwork,
00375              int *info);
00376 #else
00377
00411 void sormqr_(char *side,
00412              char *trans,
00413              int *m,
00414              int *n,
00415              int *k,
00416              Real *a,
00417              int *lda,
00418              Real *tau,
00419              Real *c,
00420              int *ldc,
00421              Real *work,
00422              int *lwork,
00423              int *info);
00424 #endif
00425 }
00426 }
00427
00428 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
                                mtk::Real *rhs) {
00429
00430
00431     #if MTK_DEBUG_LEVEL > 0
00432     mtk::Tools::Prevent(rhs == nullptr, __FILE__, __LINE__, __func__);
00433     #endif
00434
00435     int *ipiv{}; // Array for pivoting information.
00436     int nrhs{1}; // Number of right-hand sides.
00437     int info{}; // Status of the solution.
00438     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00439
00440     try {
00441         ipiv = new int[mm_rank];
00442     } catch (std::bad_alloc &memory_allocation_exception) {
00443         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```

```

00444         std::endl;
00445         std::cerr << memory_allocation_exception.what() << std::endl;
00446     }
00447     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00448
00449     int ldbb = mm_rank;
00450     int mm_ld = mm_rank;
00451
00452     #ifdef MTK_PRECISION_DOUBLE
00453     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00454     #else
00455     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00456     #endif
00457
00458     delete [] ipiv;
00459     return info;
00460 }
00461
00462 int mtk::LAPACKAdapter::SolveDenseSystem(
00463     mtk::DenseMatrix &mm,
00464                                     mtk::DenseMatrix &bb) {
00465
00466     int nrhs{bb.num_rows()}; // Number of right-hand sides.
00467
00468     #if MTK_DEBUG_LEVEL > 0
00469     mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00470     #endif
00471
00472     int *ipiv{}; // Array for pivoting information.
00473     int info{}; // Status of the solution.
00474     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00475
00476     try {
00477         ipiv = new int[mm_rank];
00478     } catch (std::bad_alloc &memory_allocation_exception) {
00479         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00480             std::endl;
00481         std::cerr << memory_allocation_exception.what() << std::endl;
00482     }
00483     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00484
00485     int ldbb = mm_rank;
00486     int mm_ld = mm_rank;
00487
00488     #ifdef MTK_PRECISION_DOUBLE
00489     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00490     #else
00491     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00492     #endif
00493
00494     delete [] ipiv;
00495
00496     // After output, the data in the matrix will be column-major ordered.
00497
00498     bb.SetOrdering(mtk::COL_MAJOR);
00499
00500     #if MTK_DEBUG_LEVEL > 0
00501     std::cout << "bb_col_maj_ord =" << std::endl;
00502     std::cout << bb << std::endl;
00503     #endif
00504
00505     bb.OrderRowMajor();
00506
00507     #if MTK_DEBUG_LEVEL > 0
00508     std::cout << "bb_row_maj_ord =" << std::endl;
00509     std::cout << bb << std::endl;
00510     #endif
00511
00512     return info;
00513 }
00514
00515 int mtk::LAPACKAdapter::SolveDenseSystem(
00516     mtk::DenseMatrix &mm,
00517                                     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.

```

```

00523
00524     try {
00525         ipiv = new int[mm_rank];
00526     } catch (std::bad_alloc &memory_allocation_exception) {
00527         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00528             std::endl;
00529         std::cerr << memory_allocation_exception.what() << std::endl;
00530     }
00531     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00532
00533     int ldbb = mm_rank;
00534     int mm_ld = mm_rank;
00535
00536     mm.OrderColMajor();
00537
00538     #ifdef MTK_PRECISION_DOUBLE
00539     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00540         rhs.discrete_field_u(), &ldbb, &info);
00541     #else
00542     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00543         rhs.discrete_field_u(), &ldbb, &info);
00544     #endif
00545
00546     mm.OrderRowMajor();
00547
00548     delete [] ipiv;
00549
00550     return info;
00551 }
00552
00553 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
(mtk::DenseMatrix &aa) {
00554
00555     mtk::Real *work{}; // Working array.
00556     mtk::Real *tau{}; // Array for the Householder scalars.
00557
00558     // Prepare to factorize: allocate and inquire for the value of lwork.
00559     try {
00560         work = new mtk::Real[1];
00561     } catch (std::bad_alloc &memory_allocation_exception) {
00562         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00563             std::endl;
00564         std::cerr << memory_allocation_exception.what() << std::endl;
00565     }
00566     memset(work, mtk::kZero, sizeof(aa.data()[0])*1);
00567
00568     int lwork{-1};
00569     int info{};
00570
00571     int aa_num_cols = aa.num_cols();
00572     int aaT_num_rows = aa.num_cols();
00573     int aaT_num_cols = aa.num_rows();
00574
00575     #if MTK_DEBUG_LEVEL > 0
00576     std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00577     std::cout << aa << std::endl;
00578     #endif
00579
00580     #ifdef MTK_PRECISION_DOUBLE
00581     dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00582         tau,
00583         work, &lwork, &info);
00584     #else
00585     fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00586         tau,
00587         work, &lwork, &info);
00588     #endif
00589
00590     #if MTK_DEBUG_LEVEL > 0
00591     if (info == 0) {
00592         lwork = (int) work[0];
00593     } else {
00594         std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00595             std::endl;
00596         std::cerr << "Exiting..." << std::endl;
00597     }
00598     #endif
00599
00600     #if MTK_DEBUG_LEVEL > 0
00601     std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00602         << std::endl;

```

```

00603 #endif
00604
00605 delete [] work;
00606 work = nullptr;
00607
00608 // Once we know lwork, we can actually invoke the factorization:
00609 try {
00610     work = new mtk::Real [lwork];
00611 } catch (std::bad_alloc &memory_allocation_exception) {
00612     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00613         std::endl;
00614     std::cerr << memory_allocation_exception.what() << std::endl;
00615 }
00616 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00617
00618 int ltau = std::min(aaT_num_rows, aaT_num_cols);
00619
00620 try {
00621     tau = new mtk::Real [ltau];
00622 } catch (std::bad_alloc &memory_allocation_exception) {
00623     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00624         std::endl;
00625     std::cerr << memory_allocation_exception.what() << std::endl;
00626 }
00627 memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00628
00629 #ifdef MTK_PRECISION_DOUBLE
00630 dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00631     tau, work, &lwork, &info);
00632 #else
00633 fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00634     tau, work, &lwork, &info);
00635 #endif
00636
00637 if (!info) {
00638     #if MTK_DEBUG_LEVEL > 0
00639         std::cout << "QR factorization completed!" << std::endl << std::endl;
00640     #endif
00641 } else {
00642     std::cerr << "Error solving system! info = " << info << std::endl;
00643     std::cerr << "Exiting..." << std::endl;
00644 }
00645
00646 #if MTK_DEBUG_LEVEL > 0
00647 std::cout << "Input matrix AFTER QR factorization:" << std::endl;
00648 std::cout << aa << std::endl;
00649 #endif
00650
00651 // We now generate the real matrix Q with orthonormal columns. This has to
00652 // be done separately since the actual output of dgeqrf_ (AA_) represents
00653 // the orthogonal matrix Q as a product of min(aa_num_rows, aa_num_cols)
00654 // elementary Householder reflectors. Notice that we must re-inquire the new
00655 // value for lwork that is used.
00656
00657 bool padded{false};
00658
00659 bool transpose{false};
00660
00661 mtk::DenseMatrix QQ_(aa.num_cols(), padded, transpose);
00662
00663 #if MTK_DEBUG_LEVEL > 0
00664 std::cout << "Initialized QQ_T: " << std::endl;
00665 std::cout << QQ_ << std::endl;
00666 #endif
00667
00668 // Assemble the QQ_ matrix:
00669 lwork = -1;
00670
00671 delete[] work;
00672 work = nullptr;
00673
00674 try {
00675     work = new mtk::Real[lwork];
00676 } catch (std::bad_alloc &memory_allocation_exception) {
00677     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00678         std::endl;
00679     std::cerr << memory_allocation_exception.what() <<
00680         std::endl;
00681 }
00682 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00683

```



```

00684 char side_{'L'};
00685 char trans_{'N'};
00686
00687 int aux = QQ_.num_rows();
00688
00689 #ifdef MTK_PRECISION_DOUBLE
00690 dormqr_(&side_, &trans_,
00691         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00692         QQ_.data(), &aux, work, &lwork, &info);
00693 #else
00694 formqr_(&side_, &trans_,
00695         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00696         QQ_.data(), &aux, work, &lwork, &info);
00697 #endif
00698
00699 #if MTK_DEBUG_LEVEL > 0
00700 if (info == 0) {
00701     lwork = (int) work[0];
00702 } else {
00703     std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00704     std::cerr << "Exiting..." << std::endl;
00705 }
00706 #endif
00707
00708 #if MTK_DEBUG_LEVEL > 0
00709 std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00710     std::endl << std::endl;
00711 #endif
00712
00713 delete[] work;
00714 work = nullptr;
00715
00716 try {
00717     work = new mtk::Real[lwork];
00718 } catch (std::bad_alloc &memory_allocation_exception) {
00719     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00720         std::endl;
00721     std::cerr << memory_allocation_exception.what() << std::endl;
00722 }
00723 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00724
00725 #ifdef MTK_PRECISION_DOUBLE
00726 dormqr_(&side_, &trans_,
00727         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00728         QQ_.data(), &aux, work, &lwork, &info);
00729 #else
00730 formqr_(&side_, &trans_,
00731         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00732         QQ_.data(), &aux, work, &lwork, &info);
00733 #endif
00734
00735 if (!info) {
00736     #if MTK_DEBUG_LEVEL>0
00737     std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00738     #endif
00739 } else {
00740     std::cerr << "Something went wrong solving system! info = " << info <<
00741         std::endl;
00742     std::cerr << "Exiting..." << std::endl;
00743 }
00744
00745 delete[] work;
00746 work = nullptr;
00747
00748 delete[] tau;
00749 tau = nullptr;
00750
00751 return QQ_;
00752 }
00753
00754 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
    mtk::DenseMatrix &aa,
                                mtk::Real *ob_,
                                int ob_ld_) {
00755
00756     // We first invoke the solver to query for the value of lwork. For this,
00757     // we must at least allocate enough space to allow access to WORK(1), or
00758     // work[0]:
00759
00760     // If LWORK = -1, then a workspace query is assumed; the routine only
00761     // calculates the optimal size of the WORK array, returns this value as

```

```

00764 // the first entry of the WORK array, and no error message related to
00765 // LWORK is issued by XERBLA.
00766
00767 mtk::Real *work{}; // Work array.
00768
00769 try {
00770     work = new mtk::Real[l];
00771 } catch (std::bad_alloc &memory_allocation_exception) {
00772     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00773     std::cerr << memory_allocation_exception.what() << std::endl;
00774 }
00775 memset(work, mtk::kZero, sizeof(work[0])*l);
00776
00777 char trans_{'N'};
00778 int nrhs_{1};
00779 int info{0};
00780 int lwork{-1};
00781
00782 int AA_num_rows_ = aa.num_cols();
00783 int AA_num_cols_ = aa.num_rows();
00784 int AA_ld_ = std::max(1, aa.num_cols());
00785
00786 #ifdef MTK_PRECISION_DOUBLE
00787 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00788     ob_, &ob_ld_,
00789     work, &lwork, &info);
00790 #else
00791 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00792     ob_, &ob_ld_,
00793     work, &lwork, &info);
00794 #endif
00795
00796 if (info == 0) {
00797     lwork = (int) work[0];
00798 } else {
00799     std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00800     std::endl;
00801     std::cerr << "Exiting..." << std::endl;
00802     return info;
00803 }
00804
00805 #if MTK_DEBUG_LEVEL > 0
00806 std::cout << "lwork = " << std::endl << std::setw(12)<< lwork <<
00807     std::endl << std::endl;
00808 #endif
00809
00810 // We then use lwork's new value to create the work array:
00811 delete[] work;
00812 work = nullptr;
00813
00814 try {
00815     work = new mtk::Real[lwork];
00816 } catch (std::bad_alloc &memory_allocation_exception) {
00817     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00818     std::cerr << memory_allocation_exception.what() << std::endl;
00819 }
00820 memset(work, 0.0, sizeof(work[0])*lwork);
00821
00822 // We now invoke the solver again:
00823 #ifdef MTK_PRECISION_DOUBLE
00824 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00825     ob_, &ob_ld_,
00826     work, &lwork, &info);
00827 #else
00828 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00829     ob_, &ob_ld_,
00830     work, &lwork, &info);
00831 #endif
00832
00833 delete [] work;
00834 work = nullptr;
00835
00836 return info;
00837 }

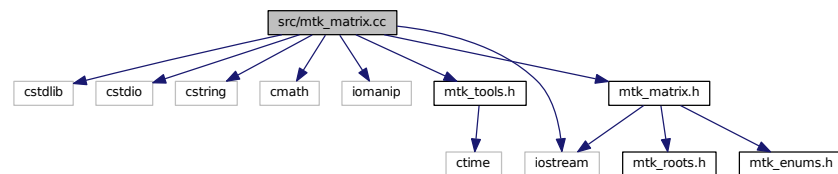
```

## 17.75 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:



### 17.75.1 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](#).

## 17.76 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 the source code should be reported to:
00018
00019 esanchez at mail dot sdsu dot edu
00020
00021 A copy of the modified files should be reported once modifications are
00022 completed. Documentation related to said modifications should be included.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page:
00026
00027 http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
```

```

00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. 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 <cstdlib>
00062 #include <cstdio>
00063 #include <cstring>
00064 #include <cmath>
00065
00066 #include <iomanip>
00067 #include <iostream>
00068
00069 #include "mtk_tools.h"
00070 #include "mtk_matrix.h"
00071
00072 mtk::Matrix::Matrix():
00073     storage_(mtk::DENSE),
00074     ordering_(mtk::ROW_MAJOR),
00075     num_rows_(),
00076     num_cols_(),
00077     num_values_(),
00078     ld_(),
00079     num_zero_(),
00080     num_non_zero_(),
00081     num_null_(),
00082     num_non_null_(),
00083     kl_(),
00084     ku_(),
00085     bandwidth_(),
00086     abs_density_(),
00087     rel_density_(),
00088     abs_sparsity_(),
00089     rel_sparsity_() {}
00090
00091 mtk::Matrix::Matrix(const Matrix &in):
00092     storage_(in.storage_),
00093     ordering_(in.ordering_),
00094     num_rows_(in.num_rows_),
00095     num_cols_(in.num_cols_),
00096     num_values_(in.num_values_),
00097     ld_(in.ld_),
00098     num_zero_(in.num_zero_),
00099     num_non_zero_(in.num_non_zero_),
00100     num_null_(in.num_null_),
00101     num_non_null_(in.num_non_null_),
00102     kl_(in.kl_),
00103     ku_(in.ku_),
00104     bandwidth_(in.bandwidth_),
00105     abs_density_(in.abs_density_),
00106     rel_density_(in.rel_density_),
00107     abs_sparsity_(in.abs_sparsity_),
00108     rel_sparsity_(in.rel_sparsity_) {}
00109
00110 mtk::Matrix::~Matrix() {}
00111
00112 mtk::MatrixStorage mtk::Matrix::storage() const {

```

```
00113
00114     return storage_;
00115 }
00116
00117 mtk::MatrixOrdering mtk::Matrix::ordering() const {
00118
00119     return ordering_;
00120 }
00121
00122 int mtk::Matrix::num_rows() const {
00123
00124     return num_rows_;
00125 }
00126
00127 int mtk::Matrix::num_cols() const {
00128
00129     return num_cols_;
00130 }
00131
00132 int mtk::Matrix::num_values() const {
00133
00134     return num_values_;
00135 }
00136
00137 int mtk::Matrix::ld() const {
00138
00139     return ld_;
00140 }
00141
00142 int mtk::Matrix::num_zero() const {
00143
00144     return num_zero_;
00145 }
00146
00147 int mtk::Matrix::num_non_zero() const {
00148
00149     return num_non_zero_;
00150 }
00151
00152 int mtk::Matrix::num_null() const {
00153
00154     return num_null_;
00155 }
00156
00157 int mtk::Matrix::num_non_null() const {
00158
00159     return num_non_null_;
00160 }
00161
00162 int mtk::Matrix::kl() const {
00163
00164     return kl_;
00165 }
00166
00167 int mtk::Matrix::ku() const {
00168
00169     return ku_;
00170 }
00171
00172 int mtk::Matrix::bandwidth() const {
00173
00174     return bandwidth_;
00175 }
00176
00177 mtk::Real mtk::Matrix::rel_density() const {
00178
00179     return rel_density_;
00180 }
00181
00182 mtk::Real mtk::Matrix::abs_sparsity() const {
00183
00184     return abs_sparsity_;
00185 }
00186
00187 mtk::Real mtk::Matrix::rel_sparsity() const {
00188
00189     return rel_sparsity_;
00190 }
00191
00192 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss) {
00193
```

```

00194     #if MTK_DEBUG_LEVEL > 0
00195     mtk::Tools::Prevent(!(ss == mtk::DENSE ||
00196                             ss == mtk::BANDED ||
00197                             ss == mtk::CRS),
00198                         __FILE__, __LINE__, __func__);
00199     #endif
00200     storage_ = ss;
00201 }
00202
00203
00204 void mtk::Matrix::set_ordering(const
    mtk::MatrixOrdering &oo) {
00205
00206     #if MTK_DEBUG_LEVEL > 0
00207     mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
    mtk::COL_MAJOR),
00208                         __FILE__, __LINE__, __func__);
00209     #endif
00210     ordering_ = oo;
00211
00212     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00213         std::max(1,num_cols_): std::max(1,num_rows_);
00214 }
00215
00216
00217 void mtk::Matrix::set_num_rows(int in) {
00218
00219     #if MTK_DEBUG_LEVEL > 0
00220     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00221     #endif
00222     num_rows_ = in;
00223     num_values_ = num_rows_*num_cols_;
00224     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00225         std::max(1,num_cols_): std::max(1,num_rows_);
00226 }
00227
00228
00229 void mtk::Matrix::set_num_cols(int in) {
00230
00231     #if MTK_DEBUG_LEVEL > 0
00232     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00233     #endif
00234     num_cols_ = in;
00235     num_values_ = num_rows_*num_cols_;
00236     ld_ = (ordering_ == mtk::ROW_MAJOR)?
00237         std::max(1,num_cols_): std::max(1,num_rows_);
00238 }
00239
00240
00241 void mtk::Matrix::set_num_zero(int in) {
00242
00243     #if MTK_DEBUG_LEVEL > 0
00244     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00245     #endif
00246     num_zero_ = in;
00247     num_non_zero_ = num_values_ - num_zero_;
00248
00249     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00250     rel_sparsity_ = 1.0 - rel_density_;
00251 }
00252
00253
00254
00255 void mtk::Matrix::set_num_null(int in) {
00256
00257     #if MTK_DEBUG_LEVEL > 0
00258     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00259     #endif
00260     num_null_ = in;
00261     num_non_null_ = num_values_ - num_null_;
00262
00263     abs_density_ = (mtk::Real) num_non_null_/num_values_;
00264     abs_sparsity_ = 1.0 - abs_density_;
00265 }
00266
00267
00268
00269 void mtk::Matrix::IncreaseNumZero() {
00270
00271     num_zero_++;
00272     num_non_zero_ = num_values_ - num_zero_;
00273     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00274 }
00275

```

```

00276   rel_sparsity_ = 1.0 - rel_density_;
00277 }
00278
00279 void mtk::Matrix::IncreaseNumNull() {
00280
00282
00283   num_null_++;
00284   num_non_null_ = num_values_ - num_null_;
00285   abs_density_ = (mtk::Real) num_non_null_/num_values_;
00286   abs_sparsity_ = 1.0 - abs_density_;
00287 }

```

## 17.77 src/mtk\_tools.cc File Reference

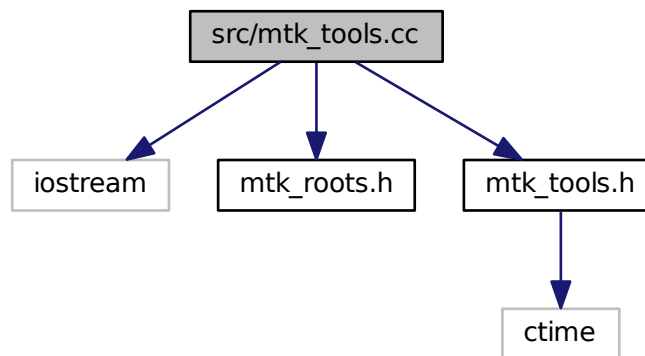
Implements a execution tool manager class.

```

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

```

Include dependency graph for mtk\_tools.cc:



### 17.77.1 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](#).

## 17.78 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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
00084
00085 int mtk::Tools::test_number_; // Used to control the correctness of the test.
00086
00087 clock_t mtk::Tools::begin_time_; // Used to time tests.
00088
00089 void mtk::Tools::BeginTestNo(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     std::cout << "Test " << nn << "..." << std::endl << std::endl;
00098     begin_time_ = clock();
00099 }
00100
00101 void mtk::Tools::EndTestNo(const int &nn) {
00102
00103     #if MTK_DEBUG_LEVEL > 0
00104     mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00105     #endif
00106
00107     auto duration = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00108     std::cout << "Test " << test_number_ << " complete! ";
00109     std::cout << "Elapsed: " << duration << " seconds." << std::endl;
00110 }

```

## 17.79 src/mtk\_uni\_stg\_grid\_1d.cc File Reference

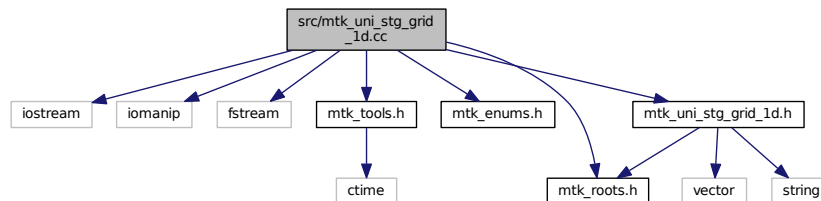
Implementation of an 1D uniform staggered grid.

```

#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)`

#### 17.79.1 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](#).

**17.80 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_enums.h"
00062 #include "mtk_tools.h"
00063
00064 #include "mtk_uni_stg_grid_1d.h"
00065
00066 namespace mtk {
00067
00068 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid1D &in) {
00069
00070     stream << '[' << in.west_bndy_x << ':' << in.num_cells_x << ':' <<
00071     in.east_bndy_x << "]" = " << std::endl << std::endl;
00072
00073
00074
00075     stream << "x:";
00076     for (unsigned int ii = 0; ii < in.discrete_domain_x.size(); ++ii) {
00077         stream << std::setw(10) << in.discrete_domain_x[ii];

```

```

00078     }
00079     stream << std::endl;
00080
00082
00083     if (in.nature_ == mtk::SCALAR) {
00084         stream << "u:";
00085     }
00086     else {
00087         stream << "v:";
00088     }
00089     for (unsigned int ii = 0; ii < in.discrete_field_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
UniStgGrid1D &grid):
00109     nature_(grid.nature_),
00110     west_bndy_x_(grid.west_bndy_x_),
00111     east_bndy_x_(grid.east_bndy_x_),
00112     num_cells_x_(grid.num_cells_x_),
00113     delta_x_(grid.delta_x_) {
00114
00115         std::copy(grid.discrete_domain_x_.begin(),
00116                 grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
00117                 discrete_domain_x_.begin());
00118
00119         std::copy(grid.discrete_field_u_.begin(),
00120                 grid.discrete_field_u_.begin() + grid.
discrete_field_u_.size(),
00121                 discrete_field_u_.begin());
00122     }
00123
00124 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00125                                 const Real &east_bndy_x,
00126                                 const int &num_cells_x,
00127                                 const mtk::FieldNature &nature) {
00128
00129     #if MTK_DEBUG_LEVEL > 0
00130     mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00131     mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00132     mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00133     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00134     #endif
00135
00136     nature_ = nature;
00137     west_bndy_x_ = west_bndy_x;
00138     east_bndy_x_ = east_bndy_x;
00139     num_cells_x_ = num_cells_x;
00140
00141     delta_x_ = (east_bndy_x - west_bndy_x) / (mtk::Real) num_cells_x;
00142 }
00143
00144 mtk::UniStgGrid1D::~~UniStgGrid1D() {}
00145
00146 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00147
00148     return west_bndy_x_;
00149 }
00150
00151 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00152
00153     return east_bndy_x_;
00154 }
00155
00156 mtk::Real mtk::UniStgGrid1D::delta_x() const {

```

```

00157
00158     return delta_x_;
00159 }
00160
00161 mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() {
00162
00163     return discrete_domain_x_.data();
00164 }
00165
00166 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
00184     discrete_domain_x_.reserve(num_cells_x_ + 2);
00185
00186     discrete_domain_x_.push_back(west_bndy_x_);
00187     #ifdef MTK_PRECISION_DOUBLE
00188     auto first_center = west_bndy_x_ + delta_x_/2.0;
00189     #else
00190     auto first_center = west_bndy_x_ + delta_x_/2.0f;
00191     #endif
00192     discrete_domain_x_.push_back(first_center);
00193     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00194         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00195     }
00196     discrete_domain_x_.push_back(east_bndy_x_);
00197
00198
00199     discrete_field_u_.reserve(num_cells_x_ + 2);
00200
00201     discrete_field_u_.push_back(ScalarField(west_bndy_x_));
00202
00203     discrete_field_u_.push_back(ScalarField(first_center));
00204     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00205         discrete_field_u_.push_back(ScalarField(first_center + ii*delta_x_));
00206     }
00207     discrete_field_u_.push_back(ScalarField(east_bndy_x_));
00208
00209 }
00210
00211 void mtk::UniStgGrid1D::BindVectorField(
00212     mtk::Real (*VectorField)(mtk::Real xx)) {
00213
00214     #if MTK_DEBUG_LEVEL > 0
00215     mtk::Tools::Prevent(nature_ == mtk::SCALAR, __FILE__, __LINE__, __func__);
00216     #endif
00217
00218
00219     discrete_domain_x_.reserve(num_cells_x_ + 1);
00220
00221     discrete_domain_x_.push_back(west_bndy_x_);
00222     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00223         discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00224     }
00225     discrete_domain_x_.push_back(east_bndy_x_);
00226
00227
00228     discrete_field_u_.reserve(num_cells_x_ + 1);
00229
00230     discrete_field_u_.push_back(VectorField(west_bndy_x_));
00231     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00232         discrete_field_u_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00233     }
00234     discrete_field_u_.push_back(VectorField(east_bndy_x_));
00235
00236 }
00237
00238 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00239                                     std::string space_name,

```

```

00242                                     std::string field_name) {
00243
00244     std::ofstream output_dat_file; // Output file.
00245     output_dat_file.open(filename);
00246
00247     if (!output_dat_file.is_open()) {
00248         return false;
00249     }
00250
00251     output_dat_file << "#" << space_name << " " << field_name << std::endl;
00252     for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00253         output_dat_file << discrete_domain_x_[ii] << " " << discrete_field_u_[ii] <<
00254             std::endl;
00255     }
00256
00257     output_dat_file.close();
00258
00259     return true;
00260 }
00261

```

## 17.81 src/mtk\_uni\_stg\_grid\_2d.cc File Reference

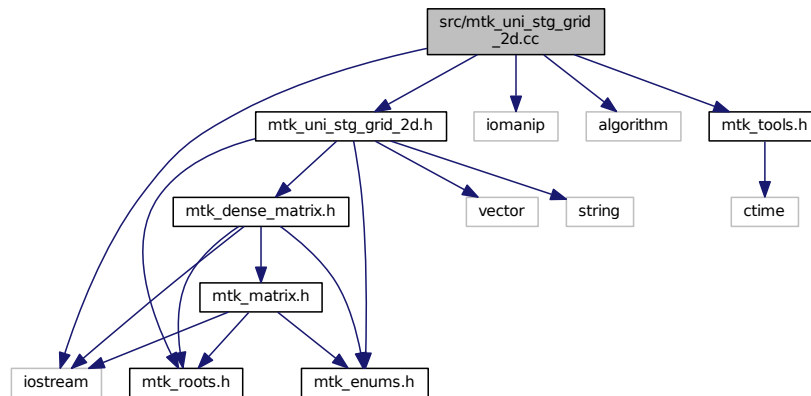
Implementation of a 2D uniform staggered grid.

```

#include <iostream>
#include <iomanip>
#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)`

### 17.81.1 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](#).

## 17.82 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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
00059 #include <algorithm>
00060
00061 #include "mtk_tools.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk {
00065
00066 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid2D &in) {
00067
00068     stream << '[' << in.west_bndy_x << ':' << in.num_cells_x << ':' <<
00069     in.east_bndy_x << "]" x ";

```

```

00070
00071 stream << '[' << in.south_bndy_y_ << ':' << in.num_cells_y_ << ':' <<
00072 in.north_bndy_y_ << "]" = " << std::endl << std::endl;
00073
00075
00076 stream << "x:";
00077 for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00078     stream << std::setw(10) << in.discrete_domain_x_[ii];
00079 }
00080 stream << std::endl;
00081
00082 stream << "y:";
00083 for (unsigned int ii = 0; ii < in.discrete_domain_y_.size(); ++ii) {
00084     stream << std::setw(10) << in.discrete_domain_y_[ii];
00085 }
00086 stream << std::endl;
00087
00089 if (in.nature_ == mtk::SCALAR) {
00090     stream << "u:";
00091 }
00092 else {
00093     stream << "v:";
00094 }
00095
00096 if (in.discrete_field_u_.size() > 0) {
00097     for (unsigned int ii = 0; ii < in.num_cells_x_ + 2; ++ii) {
00098         for (unsigned int jj = 0; jj < in.num_cells_y_ + 2; ++jj) {
00099             stream << std::setw(10) << in.discrete_field_u_[ii*in.
num_cells_y_ + jj];
00100         }
00101         stream << std::endl;
00102     }
00103 }
00104
00105 stream << std::endl;
00106
00107 return stream;
00108 }
00109 }
00110
00111 mtk::UniStgGrid2D::UniStgGrid2D():
00112     nature_(),
00113     discrete_domain_x_(),
00114     discrete_domain_y_(),
00115     discrete_field_u_(),
00116     west_bndy_x_(),
00117     east_bndy_x_(),
00118     num_cells_x_(),
00119     delta_x_(),
00120     south_bndy_y_(),
00121     north_bndy_y_(),
00122     num_cells_y_(),
00123     delta_y_() {}
00124
00125 mtk::UniStgGrid2D::UniStgGrid2D(const
UniStgGrid2D &grid):
00126     nature_(grid.nature_),
00127     west_bndy_x_(grid.west_bndy_x_),
00128     east_bndy_x_(grid.east_bndy_x_),
00129     num_cells_x_(grid.num_cells_x_),
00130     delta_x_(grid.delta_x_),
00131     south_bndy_y_(grid.south_bndy_y_),
00132     north_bndy_y_(grid.north_bndy_y_),
00133     num_cells_y_(grid.num_cells_y_),
00134     delta_y_(grid.delta_y_) {
00135
00136     std::copy(grid.discrete_domain_x_.begin(),
00137         grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
00138         discrete_domain_x_.begin());
00139
00140     std::copy(grid.discrete_domain_y_.begin(),
00141         grid.discrete_domain_y_.begin() + grid.
discrete_domain_y_.size(),
00142         discrete_domain_y_.begin());
00143
00144     std::copy(grid.discrete_field_u_.begin(),
00145         grid.discrete_field_u_.begin() + grid.
discrete_field_u_.size(),
00146         discrete_field_u_.begin());
00147 }

```

```

00148
00149 mtk::UniStgGrid2D::UniStgGrid2D(const Real &west_bndy_x,
00150                                   const Real &east_bndy_x,
00151                                   const int &num_cells_x,
00152                                   const Real &south_bndy_y,
00153                                   const Real &north_bndy_y,
00154                                   const int &num_cells_y,
00155                                   const mtk::FieldNature &nature) {
00156
00157     #if MTK_DEBUG_LEVEL > 0
00158     mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00159     mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00160     mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00161     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00162     mtk::Tools::Prevent(south_bndy_y < mtk::kZero, __FILE__, __LINE__, __func__);
00163 ;
00164     mtk::Tools::Prevent(north_bndy_y < mtk::kZero, __FILE__, __LINE__, __func__);
00165 ;
00166     mtk::Tools::Prevent(north_bndy_y <= south_bndy_y,
00167                         __FILE__, __LINE__, __func__);
00168     mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00169 #endif
00170     nature_ = nature;
00171     west_bndy_x_ = west_bndy_x;
00172     east_bndy_x_ = east_bndy_x;
00173     num_cells_x_ = num_cells_x;
00174
00175     south_bndy_y_ = south_bndy_y;
00176     north_bndy_y_ = north_bndy_y;
00177     num_cells_y_ = num_cells_y;
00178
00179     delta_x_ = (east_bndy_x - west_bndy_x)/((mtk::Real) num_cells_x);
00180     delta_y_ = (north_bndy_y - south_bndy_y)/((mtk::Real) num_cells_y);
00181 }
00182
00183 mtk::UniStgGrid2D::~~UniStgGrid2D() {}
00184
00185 mtk::Real mtk::UniStgGrid2D::west_bndy_x() const {
00186
00187     return west_bndy_x_;
00188 }
00189
00190 mtk::Real mtk::UniStgGrid2D::east_bndy_x() const {
00191
00192     return east_bndy_x_;
00193 }
00194
00195 mtk::Real mtk::UniStgGrid2D::south_bndy_y() const {
00196
00197     return south_bndy_y_;
00198 }
00199
00200 mtk::Real mtk::UniStgGrid2D::north_bndy_y() const {
00201
00202     return north_bndy_y_;
00203 }

```

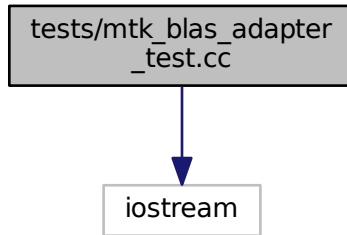
## 17.83 tests/mtk\_blas\_adapter\_test.cc File Reference

Test file for the [mtk::BLASAdapter](#) class.



```
#include <iostream>
```

Include dependency graph for mtk\_blas\_adapter\_test.cc:



## Functions

- int [main](#) ()

### 17.83.1 Detailed Description

#### Author

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

Definition in file [mtk\\_blas\\_adapter\\_test.cc](#).

### 17.83.2 Function Documentation

#### 17.83.2.1 int main ( )

Definition at line [107](#) of file [mtk\\_blas\\_adapter\\_test.cc](#).

## 17.84 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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
  
```

```

00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {
00061
00062     mtk::Tools::BeginTestNo(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     std::cout << aa << std::endl;
00077
00078     mtk::DenseMatrix bb(cc,rr);
00079
00080     bb.SetValue(0,0,7.0);
00081     bb.SetValue(0,1,8.0);
00082     bb.SetValue(1,0,9.0);
00083     bb.SetValue(1,1,10.0);
00084     bb.SetValue(2,0,11.0);
00085     bb.SetValue(2,1,12.0);
00086
00087     std::cout << bb << std::endl;
00088
00089     mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00090
00091     std::cout << pp << std::endl;
00092
00093     mtk::Tools::EndTestNo(1);
00094 }
00095
00096 int main () {
00097
00098     std::cout << "Testing mtk::BLASAdapter class." << std::endl;
00099
00100     Test1();
00101 }
00102
00103 #else
00104 #include <iostream>

```

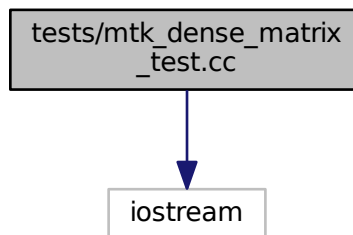
```
00105 using std::cout;
00106 using std::endl;
00107 int main () {
00108     cout << "This code HAS to be compiled with support for C++11." << endl;
00109     cout << "Exiting..." << endl;
00110 }
00111 #endif
```

## 17.85 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 ()`

#### 17.85.1 Detailed Description

##### Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)

Definition in file [mtk\\_dense\\_matrix\\_test.cc](#).

#### 17.85.2 Function Documentation

##### 17.85.2.1 `int main ( )`

Definition at line [285](#) of file [mtk\\_dense\\_matrix\\_test.cc](#).

## 17.86 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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {
00062
00063     mtk::Tools::BeginTestNo(1);
00064
00065     mtk::DenseMatrix m1;
00066
00067     std::cout << m1 << std::endl;
00068
00069     mtk::Tools::EndTestNo(1);
00070 }
00071
00072 void Test2() {
00073
00074     mtk::Tools::BeginTestNo(2);
00075
00076     int rr = 4;
00077     int cc = 7;
00078
00079     mtk::DenseMatrix m2(rr,cc);
00080
00081     std::cout << m2 << std::endl;
00082
00083     mtk::Tools::EndTestNo(2);
00084 }
00085
00086 void Test3() {
00087
00088     mtk::Tools::BeginTestNo(3);

```

```

00089
00090     int rank = 5;
00091     bool padded = true;
00092     bool transpose = false;
00093
00094     mtk::DenseMatrix m3(rank,padded,transpose);
00095
00096     std::cout << m3 << std::endl;
00097
00098     mtk::Tools::EndTestNo(3);
00099 }
00100
00101 void Test4() {
00102
00103     mtk::Tools::BeginTestNo(4);
00104
00105     int rank = 5;
00106     bool padded = false;
00107     bool transpose = false;
00108
00109     mtk::DenseMatrix m4(rank,padded,transpose);
00110
00111     std::cout << m4 << std::endl;
00112
00113     mtk::Tools::EndTestNo(4);
00114 }
00115
00116 void Test5() {
00117
00118     mtk::Tools::BeginTestNo(5);
00119
00120     int rr = 4;
00121     int cc = 7;
00122
00123     mtk::DenseMatrix m5(rr,cc);
00124
00125     for (auto ii = 0; ii < rr; ++ii) {
00126         for (auto jj = 0; jj < cc; ++jj) {
00127             m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00128         }
00129     }
00130
00131     std::cout << m5 << std::endl;
00132
00133     mtk::Real *vals = m5.data();
00134
00135     for (auto ii = 0; ii < rr; ++ii) {
00136         for (auto jj = 0; jj < cc; ++jj) {
00137             std::cout << " " << vals[ii*cc + jj];
00138         }
00139         std::cout << std::endl;
00140     }
00141     std::cout << std::endl;
00142
00143     for (auto ii = 0; ii < rr; ++ii) {
00144         for (auto jj = 0; jj < cc; ++jj) {
00145             std::cout << " " << m5.GetValue(ii,jj);
00146         }
00147         std::cout << std::endl;
00148     }
00149     std::cout << std::endl;
00150
00151     mtk::Tools::EndTestNo(5);
00152 }
00153
00154 void Test6() {
00155
00156     mtk::Tools::BeginTestNo(6);
00157
00158     bool transpose = false;
00159     int generator_length = 3;
00160     int progression_length = 4;
00161
00162     mtk::Real generator[] = {-0.5, 0.5, 1.5};
00163
00164     mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00165
00166     std::cout << m6 << std::endl;
00167
00168     transpose = true;
00169

```

```

00170     mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00171
00172     std::cout << m7 << std::endl;
00173
00174     mtk::Tools::EndTestNo(6);
00175 }
00176
00177 void Test7() {
00178
00179     mtk::Tools::BeginTestNo(7);
00180
00181     bool padded = false;
00182     bool transpose = false;
00183     int lots_of_rows = 2;
00184     int lots_of_cols = 5;
00185     mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00186
00187     std::cout << m8 << std::endl;
00188
00189     mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00190
00191     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00192         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00193             m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00194         }
00195     }
00196
00197     std::cout << m9 << std::endl;
00198
00199     mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00200
00201     std::cout << m10 << std::endl;
00202
00203     mtk::Tools::EndTestNo(7);
00204 }
00205
00206 void Test8() {
00207
00208     mtk::Tools::BeginTestNo(8);
00209
00210     int lots_of_rows = 4;
00211     int lots_of_cols = 3;
00212     mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00213
00214     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00215         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00216             m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00217         }
00218     }
00219
00220     std::cout << m11 << std::endl;
00221
00222     m11.Transpose();
00223
00224     std::cout << m11 << std::endl;
00225
00226     mtk::DenseMatrix m12;
00227
00228     m12 = m11;
00229
00230     std::cout << m12 << std::endl;
00231
00232     mtk::Tools::EndTestNo(8);
00233 }
00234
00235 void Test9() {
00236
00237     mtk::Tools::BeginTestNo(9);
00238
00239     bool transpose = false;
00240     int gg_1 = 3;
00241     int progression_length = 4;
00242     mtk::Real gg[] = {-0.5, 0.5, 1.5};
00243
00244     mtk::DenseMatrix m13(gg, gg_1 ,progression_length, transpose);
00245
00246     std::cout << m13 << std::endl;
00247
00248     mtk::DenseMatrix m14;
00249
00250

```

```

00251     m14 = m13;
00252
00253     std::cout << m14 << std::endl;
00254
00255     m13.Transpose();
00256
00257     std::cout << m13 << std::endl;
00258
00259     m14 = m13;
00260
00261     std::cout << m14 << std::endl;
00262
00263     mtk::Tools::EndTestNo(9);
00264 }
00265
00266 int main () {
00267
00268     std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00269
00270     Test1();
00271     Test2();
00272     Test3();
00273     Test4();
00274     Test5();
00275     Test6();
00276     Test7();
00277     Test8();
00278     Test9();
00279 }
00280
00281 #else
00282 #include <iostream>
00283 using std::cout;
00284 using std::endl;
00285 int main () {
00286     cout << "This code HAS to be compiled with support for C++11." << endl;
00287     cout << "Exiting..." << endl;
00288 }
00289 #endif

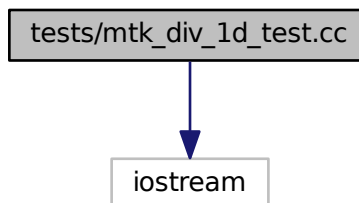
```

## 17.87 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` ()

## 17.87.1 Detailed Description

### Author

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

Definition in file [mtk\\_div\\_1d\\_test.cc](#).

## 17.87.2 Function Documentation

### 17.87.2.1 int main ( )

Definition at line 248 of file [mtk\\_div\\_1d\\_test.cc](#).

## 17.88 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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {

```



```
00061
00062     mtk::Tools::BeginTestNo(1);
00063
00064     mtk::Div1D div2;
00065
00066     bool info = div2.ConstructDiv1D();
00067
00068     if (!info) {
00069         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00070     }
00071
00072     std::cout << div2 << std::endl;
00073
00074     mtk::Tools::EndTestNo(1);
00075 }
00076
00077 void Test2() {
00078
00079     mtk::Tools::BeginTestNo(2);
00080
00081     mtk::Div1D div4;
00082
00083     bool info = div4.ConstructDiv1D(4);
00084
00085     if (!info) {
00086         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00087     }
00088
00089     std::cout << div4 << std::endl;
00090
00091     mtk::Tools::EndTestNo(2);
00092 }
00093
00094 void Test3() {
00095
00096     mtk::Tools::BeginTestNo(3);
00097
00098     mtk::Div1D div6;
00099
00100     bool info = div6.ConstructDiv1D(6);
00101
00102     if (!info) {
00103         std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00104     }
00105
00106     std::cout << div6 << std::endl;
00107
00108     mtk::Tools::EndTestNo(3);
00109 }
00110
00111 void Test4() {
00112
00113     mtk::Tools::BeginTestNo(4);
00114
00115     mtk::Div1D div8;
00116
00117     bool info = div8.ConstructDiv1D(8);
00118
00119     if (!info) {
00120         std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00121     }
00122
00123     std::cout << div8 << std::endl;
00124
00125     mtk::Tools::EndTestNo(4);
00126 }
00127
00128 void Test5() {
00129
00130     mtk::Tools::BeginTestNo(5);
00131
00132     mtk::Div1D div10;
00133
00134     bool info = div10.ConstructDiv1D(10);
00135
00136     if (!info) {
00137         std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00138     }
00139
00140     std::cout << div10 << std::endl;
00141
```

```

00142     mtk::Tools::EndTestNo(5);
00143 }
00144
00145 void Test6() {
00146     mtk::Tools::BeginTestNo(6);
00147     mtk::Div1D div12;
00148     bool info = div12.ConstructDiv1D(12);
00149     if (!info) {
00150         std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00151     }
00152     std::cout << div12 << std::endl;
00153     mtk::Tools::EndTestNo(6);
00154 }
00155
00156 void Test7() {
00157     mtk::Tools::BeginTestNo(7);
00158     mtk::Div1D div14;
00159     bool info = div14.ConstructDiv1D(14);
00160     if (!info) {
00161         std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00162     }
00163     std::cout << div14 << std::endl;
00164     mtk::Tools::EndTestNo(7);
00165 }
00166
00167 void Test8() {
00168     mtk::Tools::BeginTestNo(8);
00169     mtk::Div1D div2;
00170     bool info = div2.ConstructDiv1D();
00171     if (!info) {
00172         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00173     }
00174     std::cout << div2 << std::endl;
00175     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00176     std::cout << grid << std::endl;
00177     mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00178     std::cout << div2m << std::endl;
00179     mtk::Tools::EndTestNo(8);
00180 }
00181
00182 void Test9() {
00183     mtk::Tools::BeginTestNo(9);
00184     mtk::Div1D div4;
00185     bool info = div4.ConstructDiv1D(4);
00186     if (!info) {
00187         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00188     }
00189     std::cout << div4 << std::endl;
00190     mtk::UniStgGrid1D grid(0.0, 1.0, 11);
00191     std::cout << grid << std::endl;
00192     mtk::DenseMatrix div4m(div4.ReturnAsDenseMatrix(grid));

```

```

00223
00224     std::cout << div4m << std::endl;
00225
00226     mtk::Tools::EndTestNo(9);
00227 }
00228
00229 int main () {
00230
00231     std::cout << "Testing mtk::Div1D class." << std::endl;
00232
00233     Test1();
00234     Test2();
00235     Test3();
00236     Test4();
00237     Test5();
00238     Test6();
00239     Test7();
00240     Test8();
00241     Test9();
00242 }
00243
00244 #else
00245 #include <iostream>
00246 using std::cout;
00247 using std::endl;
00248 int main () {
00249     cout << "This code HAS to be compiled with support for C++11." << endl;
00250     cout << "Exiting..." << endl;
00251 }
00252 #endif

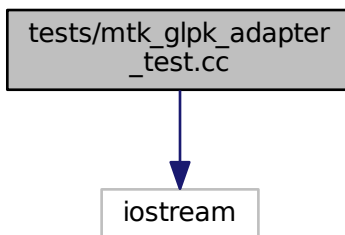
```

## 17.89 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 ()`

#### 17.89.1 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`.

## 17.89.2 Function Documentation

### 17.89.2.1 `int main ( )`

Definition at line 81 of file `mtk_glpk_adapter_test.cc`.

## 17.90 `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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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::BeginTestNo(1);
00066
00067     mtk::Tools::EndTestNo(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

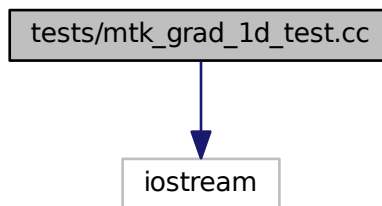
```

## 17.91 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](#) ()

### 17.91.1 Detailed Description

#### Author

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

Definition in file [mtk\\_grad\\_1d\\_test.cc](#).

## 17.91.2 Function Documentation

### 17.91.2.1 `int main ( )`

Definition at line 186 of file `mtk_grad_1d_test.cc`.

## 17.92 `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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {
00061
00062     mtk::Tools::BeginTestNo(1);
00063
00064     mtk::Grad1D grad2;
00065
00066     bool info = grad2.ConstructGrad1D();
00067
00068     if (!info) {
00069         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00070     }
00071
00072     std::cout << grad2 << std::endl;
00073

```

```
00074     mtk::Tools::EndTestNo(1);
00075 }
00076
00077 void Test2() {
00078     mtk::Tools::BeginTestNo(2);
00080     mtk::Grad1D grad4;
00082     bool info = grad4.ConstructGrad1D(4);
00084     if (!info) {
00086         std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00087     }
00088     std::cout << grad4 << std::endl;
00090     mtk::Tools::EndTestNo(2);
00092 }
00093
00094 void Test3() {
00095     mtk::Tools::BeginTestNo(3);
00097     mtk::Grad1D grad6;
00099     bool info = grad6.ConstructGrad1D(6);
00101     if (!info) {
00103         std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00104     }
00105     std::cout << grad6 << std::endl;
00107     mtk::Tools::EndTestNo(3);
00109 }
00110
00111 void Test4() {
00112     mtk::Tools::BeginTestNo(4);
00114     mtk::Grad1D grad8;
00116     bool info = grad8.ConstructGrad1D(8);
00118     if (!info) {
00120         std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00121     }
00122     std::cout << grad8 << std::endl;
00124     mtk::Tools::EndTestNo(4);
00126 }
00127
00128 void Test5() {
00129     mtk::Tools::BeginTestNo(5);
00131     mtk::Grad1D grad10;
00133     bool info = grad10.ConstructGrad1D(10);
00135     if (!info) {
00137         std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00138     }
00139     std::cout << grad10 << std::endl;
00141     mtk::Tools::EndTestNo(5);
00143 }
00144
00145 void Test6() {
00146     mtk::Tools::BeginTestNo(6);
00148     mtk::Grad1D grad2;
00150     bool info = grad2.ConstructGrad1D();
00152     if (!info) {
00153         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00154     }
```

```

00155     }
00156
00157     std::cout << grad2 << std::endl;
00158
00159     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00160
00161     std::cout << grid << std::endl;
00162
00163     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00164
00165     std::cout << grad2m << std::endl;
00166
00167     mtk::Tools::EndTestNo(6);
00168 }
00169
00170 int main () {
00171
00172     std::cout << "Testing mtk::Grad1D class." << std::endl;
00173
00174     Test1();
00175     Test2();
00176     Test3();
00177     Test4();
00178     Test5();
00179     Test6();
00180 }
00181
00182 #else
00183 #include <iostream>
00184 using std::cout;
00185 using std::endl;
00186 int main () {
00187     cout << "This code HAS to be compiled with support for C++11." << endl;
00188     cout << "Exiting..." << endl;
00189 }
00190 #endif

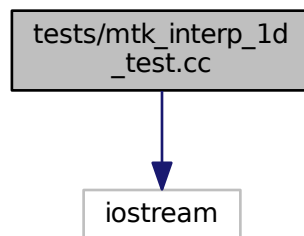
```

## 17.93 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` ()



### 17.93.1 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](#).

### 17.93.2 Function Documentation

#### 17.93.2.1 int main ( )

Definition at line 116 of file [mtk\\_interp\\_1d\\_test.cc](#).

## 17.94 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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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"
```

```

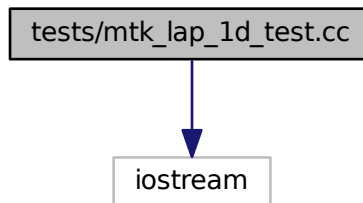
00061
00062 void Test1() {
00063
00064     mtk::Tools::BeginTestNo(1);
00065
00066     mtk::Interp1D inter;
00067
00068     bool info = inter.ConstructInterp1D();
00069
00070     if (!info) {
00071         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00072     }
00073
00074     std::cout << inter << std::endl;
00075
00076     mtk::Tools::EndTestNo(1);
00077 }
00078
00079 void Test2() {
00080
00081     mtk::Tools::BeginTestNo(2);
00082
00083     mtk::Interp1D inter;
00084
00085     bool info = inter.ConstructInterp1D();
00086
00087     if (!info) {
00088         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00089     }
00090
00091     std::cout << inter << std::endl;
00092
00093     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00094
00095     std::cout << grid << std::endl;
00096
00097     mtk::DenseMatrix interpm(inter.ReturnAsDenseMatrix(grid));
00098
00099     std::cout << interpm << std::endl;
00100
00101     mtk::Tools::EndTestNo(2);
00102 }
00103
00104 int main () {
00105
00106     std::cout << "Testing mtk::Interp1D class." << std::endl;
00107
00108     Test1();
00109     Test2();
00110 }
00111
00112 #else
00113 #include <iostream>
00114 using std::cout;
00115 using std::endl;
00116 int main () {
00117     cout << "This code HAS to be compiled with support for C++11." << endl;
00118     cout << "Exiting..." << endl;
00119 }
00120 #endif

```

## 17.95 tests/mtk\_lap\_1d\_test.cc File Reference

```
#include <iostream>
```

Include dependency graph for mtk\_lap\_1d\_test.cc:



### Functions

- int [main](#) ()

#### 17.95.1 Function Documentation

##### 17.95.1.1 int main ( )

Definition at line [156](#) of file [mtk\\_lap\\_1d\\_test.cc](#).

## 17.96 mtk\_lap\_1d\_test.cc

```
00001 #if __cplusplus == 201103L
00002
00003 #include <iostream>
00004
00005 #include "mtk.h"
00006
00007 void Test1() {
00008
00009     mtk::Tools::BeginTestNo(1);
00010
00011     mtk::Lap1D lap2;
00012
00013     bool info = lap2.ConstructLap1D();
00014
00015     if (!info) {
00016         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00017     }
00018
00019     mtk::Tools::EndTestNo(1);
00020 }
00021
00022 void Test2() {
00023
00024     mtk::Tools::BeginTestNo(2);
00025
00026     mtk::Lap1D lap4;
00027
```

```

00028     bool info = lap4.ConstructLap1D(4);
00029
00030     if (!info) {
00031         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00032     }
00033
00034     mtk::Tools::EndTestNo(2);
00035 }
00036
00037 void Test3() {
00038
00039     mtk::Tools::BeginTestNo(3);
00040
00041     mtk::Lap1D lap6;
00042
00043     bool info = lap6.ConstructLap1D(6);
00044
00045     if (!info) {
00046         std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00047     }
00048
00049     mtk::Tools::EndTestNo(3);
00050 }
00051
00052 void Test4() {
00053
00054     mtk::Tools::BeginTestNo(4);
00055
00056     mtk::Lap1D lap8;
00057
00058     bool info = lap8.ConstructLap1D(8);
00059
00060     if (!info) {
00061         std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00062     }
00063
00064     mtk::Tools::EndTestNo(4);
00065 }
00066
00067 void Test5() {
00068
00069     mtk::Tools::BeginTestNo(5);
00070
00071     mtk::Lap1D lap10;
00072
00073     bool info = lap10.ConstructLap1D(10);
00074
00075     if (!info) {
00076         std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00077     }
00078
00079     mtk::Tools::EndTestNo(5);
00080 }
00081
00082 void Test6() {
00083
00084     mtk::Tools::BeginTestNo(6);
00085
00086     mtk::Lap1D lap12;
00087
00088     bool info = lap12.ConstructLap1D(12);
00089
00090     if (!info) {
00091         std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00092     }
00093
00094     mtk::Tools::EndTestNo(6);
00095 }
00096
00097 void Test7() {
00098
00099     mtk::Tools::BeginTestNo(7);
00100
00101     mtk::Lap1D lap4;
00102
00103     bool info = lap4.ConstructLap1D(4);
00104
00105     if (!info) {
00106         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00107     }
00108 }

```

```

00109     std::cout << lap4 << std::endl;
00110     std::cout << std::endl;
00111
00112     mtk::Tools::EndTestNo(7);
00113 }
00114
00115 void Test8() {
00116     mtk::Tools::BeginTestNo(8);
00117
00118     mtk::Lap1D lap4;
00119
00120     bool info = lap4.ConstructLap1D(4);
00121
00122     if (!info) {
00123         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00124     }
00125
00126     std::cout << lap4 << std::endl;
00127     std::cout << std::endl;
00128
00129     mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00130
00131     mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00132
00133     std::cout << lap4_m << std::endl;
00134     std::cout << std::endl;
00135
00136     mtk::Tools::EndTestNo(8);
00137 }
00138
00139 int main () {
00140     std::cout << "Testing MTK 1D Laplacian" << std::endl;
00141
00142     Test1();
00143     Test2();
00144     Test3();
00145     Test4();
00146     Test5();
00147     Test6();
00148     Test7();
00149     Test8();
00150 }
00151
00152 #else
00153 #include <iostream>
00154 int main () {
00155     std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00156     std::cout << "Exiting..." << std::endl;
00157 }
00158 #endif

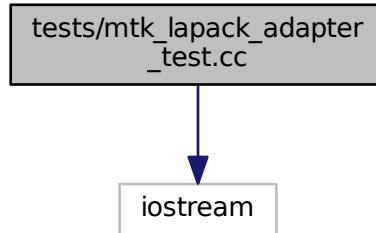
```

## 17.97 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 ()`

### 17.97.1 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`.

### 17.97.2 Function Documentation

#### 17.97.2.1 `int main ( )`

Definition at line 81 of file `mtk_lapack_adapter_test.cc`.

## 17.98 `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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. 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::BeginTestNo(1);
00066
00067     mtk::Tools::EndTestNo(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

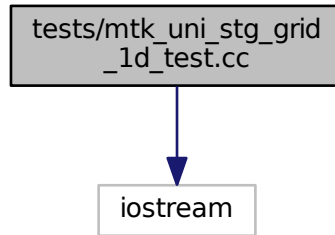
```

## 17.99 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 ()`

### 17.99.1 Detailed Description

#### Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)

Definition in file [mtk\\_uni\\_stg\\_grid\\_1d\\_test.cc](#).

### 17.99.2 Function Documentation

#### 17.99.2.1 `int main ( )`

Definition at line [164](#) of file [mtk\\_uni\\_stg\\_grid\\_1d\\_test.cc](#).

## 17.100 `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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
  
```



```

00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {
00062
00063     mtk::Tools::BeginTestNo(1);
00064
00065     mtk::UniStgGrid1D gg;
00066
00067     std::cout << gg << std::endl;
00068
00069     mtk::Tools::EndTestNo(1);
00070 }
00071
00072 mtk::Real ScalarFieldOne(mtk::Real xx) {
00073
00074     return 2.0*xx;
00075 }
00076
00077 void Test2() {
00078
00079     mtk::Tools::BeginTestNo(2);
00080
00081     mtk::Real aa = 0.0;
00082     mtk::Real bb = 1.0;
00083
00084     int nn = 5;
00085
00086     mtk::UniStgGrid1D gg(aa, bb, nn);
00087
00088     std::cout << gg << std::endl;
00089
00090     gg.BindScalarField(ScalarFieldOne);
00091
00092     std::cout << gg << std::endl;
00093
00094     mtk::Tools::EndTestNo(2);
00095 }
00096
00097 void Test3() {
00098
00099     mtk::Tools::BeginTestNo(3);
00100
00101     mtk::Real aa = 0.0;
00102     mtk::Real bb = 1.0;
00103
00104     int nn = 5;

```

```

00105
00106     mtk::UniStgGrid1D gg(aa, bb, nn);
00107
00108     std::cout << gg << std::endl;
00109
00110     gg.BindScalarField(ScalarFieldOne);
00111
00112     std::cout << gg << std::endl;
00113
00114     if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_03.dat", "x", "u(x)")) {
00115         std::cerr << "Error writing to file." << std::endl;
00116     }
00117
00118     mtk::Tools::EndTestNo(3);
00119 }
00120
00121 mtk::Real VectorFieldXComponentOne(mtk::Real xx) {
00122
00123     return xx*xx;
00124 }
00125
00126 void Test4() {
00127
00128     mtk::Tools::BeginTestNo(4);
00129
00130     mtk::Real aa = 0.0;
00131     mtk::Real bb = 1.0;
00132
00133     int nn = 20;
00134
00135     mtk::UniStgGrid1D gg(aa, bb, nn, mtk::VECTOR);
00136
00137     std::cout << gg << std::endl;
00138
00139     gg.BindVectorField(VectorFieldXComponentOne);
00140
00141     std::cout << gg << std::endl;
00142
00143     if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_04.dat", "x", "v(x)")) {
00144         std::cerr << "Error writing to file." << std::endl;
00145     }
00146
00147     mtk::Tools::EndTestNo(4);
00148 }
00149
00150 int main () {
00151
00152     std::cout << "Testing mtk::UniStgGrid1D class." << std::endl;
00153
00154     Test1();
00155     Test2();
00156     Test3();
00157     Test4();
00158 }
00159
00160 #else
00161 #include <iostream>
00162 using std::cout;
00163 using std::endl;
00164 int main () {
00165     cout << "This code HAS to be compiled with support for C++11." << endl;
00166     cout << "Exiting..." << endl;
00167 }
00168 #endif

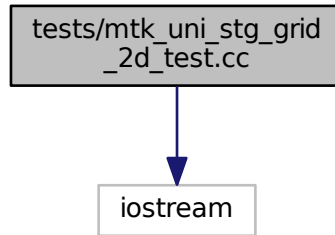
```

## 17.101 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](#) ()

### 17.101.1 Detailed Description

#### Author

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

Definition in file [mtk\\_uni\\_stg\\_grid\\_2d\\_test.cc](#).

### 17.101.2 Function Documentation

#### 17.101.2.1 int main ( )

Definition at line [102](#) of file [mtk\\_uni\\_stg\\_grid\\_2d\\_test.cc](#).

## 17.102 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. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
  
```

```

00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. 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 Test1() {
00062
00063     mtk::Tools::BeginTestNo(1);
00064
00065     mtk::UniStgGrid2D gg;
00066
00067     std::cout << gg << std::endl;
00068
00069     mtk::Tools::EndTestNo(1);
00070 }
00071
00072 void Test2() {
00073
00074     mtk::Tools::BeginTestNo(2);
00075
00076     mtk::Real aa = 0.0;
00077     mtk::Real bb = 1.0;
00078     mtk::Real cc = 0.0;
00079     mtk::Real dd = 1.0;
00080
00081     int nn = 5;
00082     int mm = 7;
00083
00084     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00085
00086     std::cout << gg << std::endl;
00087
00088     mtk::Tools::EndTestNo(2);
00089 }
00090 int main () {
00091
00092     std::cout << "Testing mtk::UniStgGrid2D class." << std::endl;
00093
00094     Test1();
00095     Test2();
00096 }
00097
00098 #else
00099 #include <iostream>
00100 using std::cout;
00101 using std::endl;
00102 int main () {
00103     cout << "This code HAS to be compiled with support for C++11." << endl;
00104     cout << "Exiting..." << endl;

```

```
00105 }  
00106 #endif
```

# Index

- ~DenseMatrix
  - mtk::DenseMatrix, [65](#)
- ~Div1D
  - mtk::Div1D, [78](#)
- ~Div2D
  - mtk::Div2D, [89](#)
- ~Grad1D
  - mtk::Grad1D, [97](#)
- ~Grad2D
  - mtk::Grad2D, [107](#)
- ~Interp1D
  - mtk::Interp1D, [110](#)
- ~Interp2D
  - mtk::Interp2D, [114](#)
- ~Lap1D
  - mtk::Lap1D, [116](#)
- ~Lap2D
  - mtk::Lap2D, [121](#)
- ~Matrix
  - mtk::Matrix, [132](#)
- ~Quad1D
  - mtk::Quad1D, [144](#)
- ~UniStgGrid1D
  - mtk::UniStgGrid1D, [152](#)
- ~UniStgGrid2D
  - mtk::UniStgGrid2D, [161](#)
- abs\_density
  - mtk::Matrix, [132](#)
- abs\_density\_
  - mtk::Matrix, [141](#)
- abs\_sparsity
  - mtk::Matrix, [132](#)
- abs\_sparsity\_
  - mtk::Matrix, [141](#)
- AssembleOperator
  - mtk::Div1D, [79](#)
  - mtk::Grad1D, [97](#)
- BANDED
  - Enumerations., [34](#)
- bandwidth
  - mtk::Matrix, [132](#)
- bandwidth\_
  - mtk::Matrix, [141](#)
- begin\_time\_
  - mtk::Tools, [148](#)
- BeginTestNo
  - mtk::Tools, [146](#)
- BindScalarField
  - mtk::UniStgGrid1D, [153](#)
  - mtk::UniStgGrid2D, [161](#)
- BindVectorField
  - mtk::UniStgGrid1D, [153](#)
- BindVectorFieldPComponent
  - mtk::UniStgGrid2D, [161](#)
- BindVectorFieldQComponent
  - mtk::UniStgGrid2D, [162](#)
- COL\_MAJOR
  - Enumerations., [34](#)
- CRS
  - Enumerations., [34](#)
- coeffs\_interior
  - mtk::Div1D, [79](#)
  - mtk::Grad1D, [97](#)
  - mtk::Interp1D, [110](#)
- coeffs\_interior\_
  - mtk::Div1D, [85](#)
  - mtk::Grad1D, [104](#)
  - mtk::Interp1D, [112](#)
- ComputePreliminaryApproximations
  - mtk::Div1D, [79](#)
  - mtk::Grad1D, [97](#)
- ComputeRationalBasisNullSpace
  - mtk::Div1D, [80](#)
  - mtk::Grad1D, [98](#)
- ComputeStencilBoundaryGrid
  - mtk::Div1D, [81](#)
  - mtk::Grad1D, [99](#)
- ComputeStencilInteriorGrid
  - mtk::Div1D, [81](#)
  - mtk::Grad1D, [99](#)
- ComputeWeights
  - mtk::Div1D, [82](#)
  - mtk::Grad1D, [100](#)
- ConstructDiv1D
  - mtk::Div1D, [82](#)
- ConstructDiv2D
  - mtk::Div2D, [89](#)
- ConstructGrad1D
  - mtk::Grad1D, [100](#)

- ConstructGrad2D
  - mtk::Grad2D, 107
- ConstructInterp1D
  - mtk::Interp1D, 110
- ConstructInterp2D
  - mtk::Interp2D, 114
- ConstructLap1D
  - mtk::Lap1D, 116
- ConstructLap2D
  - mtk::Lap2D, 121
- DENSE
  - Enumerations., 34
- Data
  - mtk::Lap1D, 117
- data
  - mtk::DenseMatrix, 65
- Data structures., 36
- data\_
  - mtk::DenseMatrix, 75
- degree\_approximation
  - mtk::Quad1D, 145
- degree\_approximation\_
  - mtk::Quad1D, 145
- delta\_x
  - mtk::UniStgGrid1D, 154
  - mtk::UniStgGrid2D, 162
- delta\_x\_
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 165
- delta\_y
  - mtk::UniStgGrid2D, 162
- delta\_y\_
  - mtk::UniStgGrid2D, 165
- DenseMatrix
  - mtk::DenseMatrix, 62–64
- dim\_null\_
  - mtk::Div1D, 85
  - mtk::Grad1D, 104
- dir\_interp\_
  - mtk::Interp1D, 112
- DirInterp
  - Enumerations., 33
- discrete\_domain\_x
  - mtk::UniStgGrid1D, 154
  - mtk::UniStgGrid2D, 162
- discrete\_domain\_x\_
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 166
- discrete\_domain\_y
  - mtk::UniStgGrid2D, 162
- discrete\_domain\_y\_
  - mtk::UniStgGrid2D, 166
- discrete\_field\_u
  - mtk::UniStgGrid1D, 154
  - mtk::UniStgGrid2D, 162
- discrete\_field\_u\_
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 166
- Div1D
  - mtk::Div1D, 78
- Div2D
  - mtk::Div2D, 89
- divergence\_
  - mtk::Div1D, 86
  - mtk::Div2D, 90
- divergence\_length\_
  - mtk::Div1D, 86
- east\_bndy\_x
  - mtk::UniStgGrid1D, 154
  - mtk::UniStgGrid2D, 162
- east\_bndy\_x\_
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 166
- EndTestNo
  - mtk::Tools, 147
- Enumerations., 33
  - BANDED, 34
  - COL\_MAJOR, 34
  - CRS, 34
  - DENSE, 34
  - DirInterp, 33
  - FieldNature, 33
  - MatrixOrdering, 33
  - MatrixStorage, 34
  - ROW\_MAJOR, 34
  - SCALAR, 33
  - SCALAR\_TO\_VECTOR, 33
  - VECTOR, 33
  - VECTOR\_TO\_SCALAR, 33
- examples/minimalistic\_poisson\_1d/minimalistic\_poisson-  
\_1d.cc, 169, 170
- examples/poisson\_1d/poisson\_1d.cc, 172, 173
- Execution tools., 35
- FieldNature
  - Enumerations., 33
- GetValue
  - mtk::DenseMatrix, 66
- Grad1D
  - mtk::Grad1D, 96
- Grad2D
  - mtk::Grad2D, 107
- gradient\_
  - mtk::Grad1D, 104
  - mtk::Grad2D, 108
- gradient\_length\_
  - mtk::Grad1D, 104
  - mtk::Grad2D, 108

- mtk::Grad1D, 104
- Grids., 38
- ImposeOnGrid
  - mtk::BCDesc1D, 51
- ImposeOnOperator
  - mtk::BCDesc1D, 52
- include/mtk.h, 176, 177
- include/mtk\_bc\_desc\_1d.h, 178, 179
- include/mtk\_blas\_adapter.h, 180, 181
- include/mtk\_dense\_matrix.h, 182, 184
- include/mtk\_div\_1d.h, 185, 187
- include/mtk\_div\_2d.h, 188, 190
- include/mtk\_enums.h, 191, 192
- include/mtk\_glpk\_adapter.h, 193, 194
- include/mtk\_grad\_1d.h, 195, 196
- include/mtk\_grad\_2d.h, 198, 199
- include/mtk\_interp\_1d.h, 200, 202
- include/mtk\_interp\_2d.h, 203, 205
- include/mtk\_lap\_1d.h, 206, 207
- include/mtk\_lap\_2d.h, 208, 210
- include/mtk\_lapack\_adapter.h, 211, 212
- include/mtk\_matrix.h, 213, 214
- include/mtk\_quad\_1d.h, 216, 218
- include/mtk\_roots.h, 219, 220
- include/mtk\_tools.h, 221, 222
- include/mtk\_uni\_stg\_grid\_1d.h, 223, 224
- include/mtk\_uni\_stg\_grid\_2d.h, 226, 227
- IncreaseNumNull
  - mtk::Matrix, 132
- IncreaseNumZero
  - mtk::Matrix, 132
- Integrate
  - mtk::Quad1D, 145
- Interp1D
  - mtk::Interp1D, 110
- Interp2D
  - mtk::Interp2D, 114
- interpolator\_
  - mtk::Interp2D, 114
- kCriticalOrderAccuracyDiv
  - Roots., 32
- kCriticalOrderAccuracyGrad
  - Roots., 32
- kDefaultMimeticThreshold
  - Roots., 32
- kDefaultOrderAccuracy
  - Roots., 32
- kDefaultTolerance
  - Roots., 32
- kOne
  - Roots., 32
- kZero
  - Roots., 32

- kl
  - mtk::Matrix, 133
- kl\_
  - mtk::Matrix, 141
- Kron
  - mtk::DenseMatrix, 67
- ku
  - mtk::Matrix, 133
- ku\_
  - mtk::Matrix, 141
- Lap1D
  - mtk::Lap1D, 116
- Lap2D
  - mtk::Lap2D, 121
- laplacian\_
  - mtk::Lap1D, 119
  - mtk::Lap2D, 121
- laplacian\_length\_
  - mtk::Lap1D, 119
- ld
  - mtk::Matrix, 133
- ld\_
  - mtk::Matrix, 141
- main
  - minimalistic\_poisson\_1d.cc, 170
  - mtk\_blas\_adapter\_test.cc, 325
  - mtk\_dense\_matrix\_test.cc, 327
  - mtk\_div\_1d\_test.cc, 332
  - mtk\_glpk\_adapter\_test.cc, 336
  - mtk\_grad\_1d\_test.cc, 338
  - mtk\_interp\_1d\_test.cc, 341
  - mtk\_lap\_1d\_test.cc, 343
  - mtk\_lapack\_adapter\_test.cc, 346
  - mtk\_uni\_stg\_grid\_1d\_test.cc, 348
  - mtk\_uni\_stg\_grid\_2d\_test.cc, 351
  - poisson\_1d.cc, 173
- Makefile.inc, 228
- Matrix
  - mtk::Matrix, 130
- matrix\_properties
  - mtk::DenseMatrix, 68
- matrix\_properties\_
  - mtk::DenseMatrix, 75
- MatrixOrdering
  - Enumerations., 33
- MatrixStorage
  - Enumerations., 34
- mim\_bndy
  - mtk::Div1D, 83
  - mtk::Grad1D, 101
- mim\_bndy\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 104



- Mimetic operators., 39
- mimetic\_threshold\_
  - mtk::Div1D, 86
  - mtk::Div2D, 90
  - mtk::Grad1D, 104
  - mtk::Grad2D, 108
  - mtk::Interp2D, 114
  - mtk::Lap1D, 119
  - mtk::Lap2D, 121
- minimalistic\_poisson\_1d.cc
  - main, 170
- minrow\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 104
- mtk, 41
  - operator<<, 43, 44
  - saxpy\_, 44
  - sgels\_, 45
  - sgemm\_, 46
  - sgemv\_, 46
  - sgeqrf\_, 47
  - sgesv\_, 47
  - snrm2\_, 47
  - sormqr\_, 48
- mtk::BCDesc1D, 51
  - ImposeOnGrid, 51
  - ImposeOnOperator, 52
- mtk::BLASAdapter, 53
  - RealAXPY, 54
  - RealDenseMM, 54
  - RealDenseMV, 56
  - RealNRM2, 58
  - RelNorm2Error, 59
- mtk::DenseMatrix, 59
  - ~DenseMatrix, 65
  - data, 65
  - data\_, 75
  - DenseMatrix, 62–64
  - GetValue, 66
  - Kron, 67
  - matrix\_properties, 68
  - matrix\_properties\_, 75
  - num\_cols, 69
  - num\_rows, 69
  - operator<<, 74
  - operator=, 70
  - OrderColMajor, 71
  - OrderRowMajor, 72
  - SetOrdering, 72
  - SetValue, 73
  - Transpose, 74
- mtk::Div1D, 75
  - ~Div1D, 78
  - AssembleOperator, 79
  - coeffs\_interior, 79
  - coeffs\_interior\_, 85
  - ComputePreliminaryApproximations, 79
  - ComputeRationalBasisNullSpace, 80
  - ComputeStencilBoundaryGrid, 81
  - ComputeStencilInteriorGrid, 81
  - ComputeWeights, 82
  - ConstructDiv1D, 82
  - dim\_null\_, 85
  - Div1D, 78
  - divergence\_, 86
  - divergence\_length\_, 86
  - mim\_bndy, 83
  - mim\_bndy\_, 86
  - mimetic\_threshold\_, 86
  - minrow\_, 86
  - num\_bndy\_coeffs, 84
  - num\_bndy\_coeffs\_, 86
  - operator<<, 85
  - order\_accuracy\_, 86
  - prem\_apps\_, 86
  - rat\_basis\_null\_space\_, 86
  - ReturnAsDenseMatrix, 84
  - row\_, 86
  - weights\_cbs, 85
  - weights\_cbs\_, 86
  - weights\_crs, 85
  - weights\_crs\_, 87
- mtk::Div2D, 87
  - ~Div2D, 89
  - ConstructDiv2D, 89
  - Div2D, 89
  - divergence\_, 90
  - mimetic\_threshold\_, 90
  - order\_accuracy\_, 90
  - ReturnAsDenseMatrix, 90
- mtk::GLPKAdapter, 91
  - SolveSimplexAndCompare, 92
- mtk::Grad1D, 93
  - ~Grad1D, 97
  - AssembleOperator, 97
  - coeffs\_interior, 97
  - coeffs\_interior\_, 104
  - ComputePreliminaryApproximations, 97
  - ComputeRationalBasisNullSpace, 98
  - ComputeStencilBoundaryGrid, 99
  - ComputeStencilInteriorGrid, 99
  - ComputeWeights, 100
  - ConstructGrad1D, 100
  - dim\_null\_, 104
  - Grad1D, 96
  - gradient\_, 104
  - gradient\_length\_, 104
  - mim\_bndy, 101

- mim\_bndy\_, 104
- mimetic\_threshold\_, 104
- minrow\_, 104
- num\_bndy\_approxs\_, 105
- num\_bndy\_coeffs\_, 102
- num\_bndy\_coeffs\_, 105
- operator<<, 104
- order\_accuracy\_, 105
- prem\_apps\_, 105
- rat\_basis\_null\_space\_, 105
- ReturnAsDenseMatrix, 102, 103
- row\_, 105
- weights\_cbs, 103
- weights\_cbs\_, 105
- weights\_crs, 103
- weights\_crs\_, 105
- mtk::Grad2D, 105
  - ~Grad2D, 107
  - ConstructGrad2D, 107
  - Grad2D, 107
  - gradient\_, 108
  - mimetic\_threshold\_, 108
  - order\_accuracy\_, 108
  - ReturnAsDenseMatrix, 108
- mtk::Interp1D, 109
  - ~Interp1D, 110
  - coeffs\_interior, 110
  - coeffs\_interior\_, 112
  - ConstructInterp1D, 110
  - dir\_interp\_, 112
  - Interp1D, 110
  - operator<<, 111
  - order\_accuracy\_, 112
  - ReturnAsDenseMatrix, 111
- mtk::Interp2D, 112
  - ~Interp2D, 114
  - ConstructInterp2D, 114
  - Interp2D, 114
  - interpolator\_, 114
  - mimetic\_threshold\_, 114
  - order\_accuracy\_, 114
  - ReturnAsDenseMatrix, 114
- mtk::LAPACKAdapter, 122
  - QRFactorDenseMatrix, 123
  - SolveDenseSystem, 124–126
  - SolveRectangularDenseSystem, 126
- mtk::Lap1D, 115
  - ~Lap1D, 116
  - ConstructLap1D, 116
  - Data, 117
  - Lap1D, 116
  - laplacian\_, 119
  - laplacian\_length\_, 119
  - mimetic\_threshold\_, 119
  - operator<<, 119
  - order\_accuracy\_, 119
  - ReturnAsDenseMatrix, 118
- mtk::Lap2D, 119
  - ~Lap2D, 121
  - ConstructLap2D, 121
  - Lap2D, 121
  - laplacian\_, 121
  - mimetic\_threshold\_, 121
  - order\_accuracy\_, 121
  - ReturnAsDenseMatrix, 121
- mtk::Matrix, 127
  - ~Matrix, 132
  - abs\_density, 132
  - abs\_density\_, 141
  - abs\_sparsity, 132
  - abs\_sparsity\_, 141
  - bandwidth, 132
  - bandwidth\_, 141
  - IncreaseNumNull, 132
  - IncreaseNumZero, 132
  - kl, 133
  - kl\_, 141
  - ku, 133
  - ku\_, 141
  - ld, 133
  - ld\_, 141
  - Matrix, 130
  - num\_cols, 133
  - num\_cols\_, 141
  - num\_non\_null, 134
  - num\_non\_null\_, 141
  - num\_non\_zero, 134
  - num\_non\_zero\_, 142
  - num\_null, 134
  - num\_null\_, 142
  - num\_rows, 134
  - num\_rows\_, 142
  - num\_values, 135
  - num\_values\_, 142
  - num\_zero, 135
  - num\_zero\_, 142
  - ordering, 135
  - ordering\_, 142
  - rel\_density, 136
  - rel\_density\_, 142
  - rel\_sparsity, 136
  - rel\_sparsity\_, 142
  - set\_num\_cols, 136
  - set\_num\_null, 137
  - set\_num\_rows, 138
  - set\_num\_zero, 138
  - set\_ordering, 139
  - set\_storage, 140

- storage, 140
- storage\_, 142
- mtk::Quad1D, 143
  - ~Quad1D, 144
  - degree\_approximation, 145
  - degree\_approximation\_, 145
  - Integrate, 145
  - operator<<, 145
  - Quad1D, 144
  - weights, 145
  - weights\_, 145
- mtk::Tools, 145
  - begin\_time\_, 148
  - BeginTestNo, 146
  - EndTestNo, 147
  - Prevent, 147
  - test\_number\_, 149
- mtk::UniStgGrid1D, 149
  - ~UniStgGrid1D, 152
  - BindScalarField, 153
  - BindVectorField, 153
  - delta\_x, 154
  - delta\_x\_, 156
  - discrete\_domain\_x, 154
  - discrete\_domain\_x\_, 156
  - discrete\_field\_u, 154
  - discrete\_field\_u\_, 156
  - east\_bndy\_x, 154
  - east\_bndy\_x\_, 156
  - nature\_, 156
  - num\_cells\_x, 155
  - num\_cells\_x\_, 156
  - operator<<, 156
  - UniStgGrid1D, 152
  - west\_bndy\_x, 155
  - west\_bndy\_x\_, 156
  - WriteToFile, 155
- mtk::UniStgGrid2D, 157
  - ~UniStgGrid2D, 161
  - BindScalarField, 161
  - BindVectorFieldPComponent, 161
  - BindVectorFieldQComponent, 162
  - delta\_x, 162
  - delta\_x\_, 165
  - delta\_y, 162
  - delta\_y\_, 165
  - discrete\_domain\_x, 162
  - discrete\_domain\_x\_, 166
  - discrete\_domain\_y, 162
  - discrete\_domain\_y\_, 166
  - discrete\_field\_u, 162
  - discrete\_field\_u\_, 166
  - east\_bndy\_x, 162
  - east\_bndy\_x\_, 166
- nature\_, 166
- north\_bndy\_y, 163
- north\_bndy\_y\_, 166
- num\_cells\_x, 163
- num\_cells\_x\_, 166
- num\_cells\_y, 163
- num\_cells\_y\_, 166
- operator<<, 165
- south\_bndy\_y, 164
- south\_bndy\_y\_, 166
- UniStgGrid2D, 160
- west\_bndy\_x, 164
- west\_bndy\_x\_, 166
- WriteToFile, 165
- mtk\_blas\_adapter\_test.cc
  - main, 325
- mtk\_dense\_matrix\_test.cc
  - main, 327
- mtk\_div\_1d\_test.cc
  - main, 332
- mtk\_glpk\_adapter\_test.cc
  - main, 336
- mtk\_grad\_1d\_test.cc
  - main, 338
- mtk\_interp\_1d\_test.cc
  - main, 341
- mtk\_lap\_1d\_test.cc
  - main, 343
- mtk\_lapack\_adapter\_test.cc
  - main, 346
- mtk\_uni\_stg\_grid\_1d\_test.cc
  - main, 348
- mtk\_uni\_stg\_grid\_2d\_test.cc
  - main, 351
- nature\_
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 166
- north\_bndy\_y
  - mtk::UniStgGrid2D, 163
- north\_bndy\_y\_
  - mtk::UniStgGrid2D, 166
- num\_bndy\_approx\_
  - mtk::Grad1D, 105
- num\_bndy\_coeffs
  - mtk::Div1D, 84
  - mtk::Grad1D, 102
- num\_bndy\_coeffs\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 105
- num\_cells\_x
  - mtk::UniStgGrid1D, 155
  - mtk::UniStgGrid2D, 163
- num\_cells\_x\_
  - mtk::UniStgGrid1D, 155
  - mtk::UniStgGrid2D, 163

- mtk::UniStgGrid1D, 156
- mtk::UniStgGrid2D, 166
- num\_cells\_y
  - mtk::UniStgGrid2D, 163
- num\_cells\_y\_
  - mtk::UniStgGrid2D, 166
- num\_cols
  - mtk::DenseMatrix, 69
  - mtk::Matrix, 133
- num\_cols\_
  - mtk::Matrix, 141
- num\_non\_null
  - mtk::Matrix, 134
- num\_non\_null\_
  - mtk::Matrix, 141
- num\_non\_zero
  - mtk::Matrix, 134
- num\_non\_zero\_
  - mtk::Matrix, 142
- num\_null
  - mtk::Matrix, 134
- num\_null\_
  - mtk::Matrix, 142
- num\_rows
  - mtk::DenseMatrix, 69
  - mtk::Matrix, 134
- num\_rows\_
  - mtk::Matrix, 142
- num\_values
  - mtk::Matrix, 135
- num\_values\_
  - mtk::Matrix, 142
- num\_zero
  - mtk::Matrix, 135
- num\_zero\_
  - mtk::Matrix, 142
- Numerical methods., 37
- operator<<
  - mtk, 43, 44
  - mtk::DenseMatrix, 74
  - mtk::Div1D, 85
  - mtk::Grad1D, 104
  - mtk::Interp1D, 111
  - mtk::Lap1D, 119
  - mtk::Quad1D, 145
  - mtk::UniStgGrid1D, 156
  - mtk::UniStgGrid2D, 165
- operator=
  - mtk::DenseMatrix, 70
- order\_accuracy\_
  - mtk::Div1D, 86
  - mtk::Div2D, 90
  - mtk::Grad1D, 105
  - mtk::Grad2D, 108
  - mtk::Interp1D, 112
  - mtk::Interp2D, 114
  - mtk::Lap1D, 119
  - mtk::Lap2D, 121
- OrderColMajor
  - mtk::DenseMatrix, 71
- OrderRowMajor
  - mtk::DenseMatrix, 72
- ordering
  - mtk::Matrix, 135
- ordering\_
  - mtk::Matrix, 142
- poisson\_1d.cc
  - main, 173
- prem\_apps\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 105
- Prevent
  - mtk::Tools, 147
- QRFactorDenseMatrix
  - mtk::LAPACKAdapter, 123
- Quad1D
  - mtk::Quad1D, 144
- ROW\_MAJOR
  - Enumerations., 34
- README.md, 231
- rat\_basis\_null\_space\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 105
- Real
  - Roots., 32
- RealAXPY
  - mtk::BLASAdapter, 54
- RealDenseMM
  - mtk::BLASAdapter, 54
- RealDenseMV
  - mtk::BLASAdapter, 56
- RealNRM2
  - mtk::BLASAdapter, 58
- rel\_density
  - mtk::Matrix, 136
- rel\_density\_
  - mtk::Matrix, 142
- rel\_sparsity
  - mtk::Matrix, 136
- rel\_sparsity\_
  - mtk::Matrix, 142
- RelNorm2Error
  - mtk::BLASAdapter, 59
- ReturnAsDenseMatrix
  - mtk::Div1D, 84

- mtk::Div2D, 90
- mtk::Grad1D, 102, 103
- mtk::Grad2D, 108
- mtk::Interp1D, 111
- mtk::Interp2D, 114
- mtk::Lap1D, 118
- mtk::Lap2D, 121
- Roots., 31
  - kCriticalOrderAccuracyDiv, 32
  - kCriticalOrderAccuracyGrad, 32
  - kDefaultMimeticThreshold, 32
  - kDefaultOrderAccuracy, 32
  - kDefaultTolerance, 32
  - kOne, 32
  - kZero, 32
  - Real, 32
- row\_
  - mtk::Div1D, 86
  - mtk::Grad1D, 105
- SCALAR
  - Enumerations., 33
- SCALAR\_TO\_VECTOR
  - Enumerations., 33
- saxpy\_
  - mtk, 44
- set\_num\_cols
  - mtk::Matrix, 136
- set\_num\_null
  - mtk::Matrix, 137
- set\_num\_rows
  - mtk::Matrix, 138
- set\_num\_zero
  - mtk::Matrix, 138
- set\_ordering
  - mtk::Matrix, 139
- set\_storage
  - mtk::Matrix, 140
- SetOrdering
  - mtk::DenseMatrix, 72
- SetValue
  - mtk::DenseMatrix, 73
- sgels\_
  - mtk, 45
- sgemm\_
  - mtk, 46
- sgemv\_
  - mtk, 46
- sgeqrf\_
  - mtk, 47
- sgesv\_
  - mtk, 47
- snrm2\_
  - mtk, 47
- SolveDenseSystem
  - mtk::LAPACKAdapter, 124–126
- SolveRectangularDenseSystem
  - mtk::LAPACKAdapter, 126
- SolveSimplexAndCompare
  - mtk::GLPKAdapter, 92
- sormqr\_
  - mtk, 48
- south\_bndy\_y
  - mtk::UniStgGrid2D, 164
- south\_bndy\_y\_
  - mtk::UniStgGrid2D, 166
- src/mtk\_bc\_desc\_1d.cc, 233
- src/mtk\_blas\_adapter.cc, 234, 235
- src/mtk\_dense\_matrix.cc, 238, 239
- src/mtk\_div\_1d.cc, 246, 247
- src/mtk\_div\_2d.cc, 263, 264
- src/mtk\_glpk\_adapter.cc, 266, 267
- src/mtk\_grad\_1d.cc, 271, 272
- src/mtk\_grad\_2d.cc, 290, 291
- src/mtk\_interp\_1d.cc, 292, 293
- src/mtk\_lap\_1d.cc, 295, 296
- src/mtk\_lap\_2d.cc, 300, 301
- src/mtk\_lapack\_adapter.cc, 302, 303
- src/mtk\_matrix.cc, 311
- src/mtk\_tools.cc, 315
- src/mtk\_uni\_stg\_grid\_1d.cc, 317, 318
- src/mtk\_uni\_stg\_grid\_2d.cc, 321, 322
- storage
  - mtk::Matrix, 140
- storage\_
  - mtk::Matrix, 142
- test\_number\_
  - mtk::Tools, 149
- tests/mtk\_blas\_adapter\_test.cc, 324, 325
- tests/mtk\_dense\_matrix\_test.cc, 327
- tests/mtk\_div\_1d\_test.cc, 331, 332
- tests/mtk\_glpk\_adapter\_test.cc, 335, 336
- tests/mtk\_grad\_1d\_test.cc, 337, 338
- tests/mtk\_interp\_1d\_test.cc, 340, 341
- tests/mtk\_lap\_1d\_test.cc, 343
- tests/mtk\_lapack\_adapter\_test.cc, 345, 346
- tests/mtk\_uni\_stg\_grid\_1d\_test.cc, 347, 348
- tests/mtk\_uni\_stg\_grid\_2d\_test.cc, 350, 351
- Transpose
  - mtk::DenseMatrix, 74
- UniStgGrid1D
  - mtk::UniStgGrid1D, 152
- UniStgGrid2D
  - mtk::UniStgGrid2D, 160
- VECTOR
  - Enumerations., 33

## VECTOR\_TO\_SCALAR

Enumerations., [33](#)

## weights

mtk::Quad1D, [145](#)

## weights\_

mtk::Quad1D, [145](#)

## weights\_cbs

mtk::Div1D, [85](#)

mtk::Grad1D, [103](#)

## weights\_cbs\_

mtk::Div1D, [86](#)

mtk::Grad1D, [105](#)

## weights\_crs

mtk::Div1D, [85](#)

mtk::Grad1D, [103](#)

## weights\_crs\_

mtk::Div1D, [87](#)

mtk::Grad1D, [105](#)

## west\_bndy\_x

mtk::UniStgGrid1D, [155](#)

mtk::UniStgGrid2D, [164](#)

## west\_bndy\_x\_

mtk::UniStgGrid1D, [156](#)

mtk::UniStgGrid2D, [166](#)

## WriteToFile

mtk::UniStgGrid1D, [155](#)

mtk::UniStgGrid2D, [165](#)