

MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.8

Mon Jul 4 2016 09:18:11

Contents

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

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

15.7.2.3	CoefficientFunction2D	43
16	Namespace Documentation	45
16.1	mtk Namespace Reference	45
16.1.1	Function Documentation	48
16.1.1.1	operator<<	48
16.1.1.2	operator<<	48
16.1.1.3	operator<<	48
16.1.1.4	operator<<	48
16.1.1.5	operator<<	49
16.1.1.6	operator<<	49
16.1.1.7	operator<<	49
16.1.1.8	operator<<	49
16.1.1.9	saxpy_	50
16.1.1.10	sgels_	50
16.1.1.11	sgemm_	51
16.1.1.12	sgemv_	52
16.1.1.13	sgeqr_	52
16.1.1.14	sgesv_	53
16.1.1.15	snrm2_	53
16.1.1.16	sormqr_	53
17	Class Documentation	55
17.1	mtk::BLASAdapter Class Reference	55
17.1.1	Detailed Description	56
17.1.2	Member Function Documentation	56
17.1.2.1	RealAXPY	56
17.1.2.2	RealDenseMM	57
17.1.2.3	RealDenseMV	59
17.1.2.4	RealDenseMV	61
17.1.2.5	RealDenseSM	61
17.1.2.6	RealNRM2	63
17.1.2.7	RelNorm2Error	64
17.2	mtk::Curl2D Class Reference	65
17.2.1	Detailed Description	67
17.2.2	Constructor & Destructor Documentation	67
17.2.2.1	Curl2D	67
17.2.2.2	Curl2D	67

17.2.2.3	<code>~Curl2D</code>	67
17.2.3	Member Function Documentation	68
17.2.3.1	<code>ConstructCurl2D</code>	68
17.2.3.2	<code>operator*</code>	68
17.2.3.3	<code>ReturnAsDenseMatrix</code>	68
17.2.4	Member Data Documentation	69
17.2.4.1	<code>curl_</code>	69
17.2.4.2	<code>mimetic_threshold_</code>	69
17.2.4.3	<code>order_accuracy_</code>	69
17.3	mtk::DenseMatrix Class Reference	69
17.3.1	Detailed Description	72
17.3.2	Constructor & Destructor Documentation	72
17.3.2.1	<code>DenseMatrix</code>	72
17.3.2.2	<code>DenseMatrix</code>	72
17.3.2.3	<code>DenseMatrix</code>	73
17.3.2.4	<code>DenseMatrix</code>	74
17.3.2.5	<code>DenseMatrix</code>	74
17.3.2.6	<code>~DenseMatrix</code>	75
17.3.3	Member Function Documentation	75
17.3.3.1	<code>data</code>	75
17.3.3.2	<code>encoded_operator</code>	76
17.3.3.3	<code>GetValue</code>	77
17.3.3.4	<code>Kron</code>	78
17.3.3.5	<code>matrix_properties</code>	79
17.3.3.6	<code>num_cols</code>	80
17.3.3.7	<code>num_rows</code>	81
17.3.3.8	<code>operator=</code>	82
17.3.3.9	<code>operator==</code>	83
17.3.3.10	<code>OrderColMajor</code>	84
17.3.3.11	<code>OrderRowMajor</code>	84
17.3.3.12	<code>set_encoded_operator</code>	85
17.3.3.13	<code>SetOrdering</code>	86
17.3.3.14	<code>SetValue</code>	87
17.3.3.15	<code>Transpose</code>	89
17.3.3.16	<code>WriteToFile</code>	90
17.3.4	Friends And Related Function Documentation	91
17.3.4.1	<code>operator<<</code>	91

17.3.5 Member Data Documentation	91
17.3.5.1 data_	91
17.3.5.2 matrix_properties_	91
17.4 mtk::Div1D Class Reference	91
17.4.1 Detailed Description	94
17.4.2 Constructor & Destructor Documentation	95
17.4.2.1 Div1D	95
17.4.2.2 Div1D	95
17.4.2.3 ~Div1D	95
17.4.3 Member Function Documentation	95
17.4.3.1 AssembleOperator	95
17.4.3.2 coeffs_interior	95
17.4.3.3 ComputePreliminaryApproximations	95
17.4.3.4 ComputeRationalBasisNullSpace	96
17.4.3.5 ComputeStencilBoundaryGrid	97
17.4.3.6 ComputeStencilInteriorGrid	97
17.4.3.7 ComputeWeights	98
17.4.3.8 ConstructDiv1D	99
17.4.3.9 mim_bndy	100
17.4.3.10 mimetic_measure	101
17.4.3.11 num_bndy_coeffs	101
17.4.3.12 num_feasible_sols	101
17.4.3.13 ReturnAsDenseMatrix	101
17.4.3.14 ReturnAsDimensionlessDenseMatrix	103
17.4.3.15 sums_rows_mim_bndy	103
17.4.3.16 weights_cbs	103
17.4.3.17 weights_crs	104
17.4.4 Friends And Related Function Documentation	104
17.4.4.1 operator<<	104
17.4.5 Member Data Documentation	104
17.4.5.1 coeffs_interior_	104
17.4.5.2 dim_null_	104
17.4.5.3 divergence_	104
17.4.5.4 divergence_length_	104
17.4.5.5 mim_bndy_	104
17.4.5.6 mimetic_measure_	104
17.4.5.7 mimetic_threshold_	105

17.4.5.8 minrow_	105
17.4.5.9 num_bndy_coeffs_	105
17.4.5.10 num_feasible_sols_	105
17.4.5.11 order_accuracy_	105
17.4.5.12 prem_apps_	105
17.4.5.13 rat_basis_null_space_	105
17.4.5.14 row_	105
17.4.5.15 sums_rows_mim_bndy_	105
17.4.5.16 weights_cbs_	105
17.4.5.17 weights_crs_	105
17.5 mtk::Div2D Class Reference	106
17.5.1 Detailed Description	108
17.5.2 Constructor & Destructor Documentation	108
17.5.2.1 Div2D	108
17.5.2.2 Div2D	108
17.5.2.3 ~Div2D	108
17.5.3 Member Function Documentation	109
17.5.3.1 ConstructDiv2D	109
17.5.3.2 ReturnAsDenseMatrix	110
17.5.4 Member Data Documentation	110
17.5.4.1 divergence_	110
17.5.4.2 mimetic_threshold_	110
17.5.4.3 order_accuracy_	110
17.6 mtk::Div3D Class Reference	110
17.6.1 Detailed Description	112
17.6.2 Constructor & Destructor Documentation	112
17.6.2.1 Div3D	112
17.6.2.2 Div3D	112
17.6.2.3 ~Div3D	112
17.6.3 Member Function Documentation	113
17.6.3.1 ConstructDiv3D	113
17.6.3.2 ReturnAsDenseMatrix	114
17.6.4 Member Data Documentation	114
17.6.4.1 divergence_	114
17.6.4.2 mimetic_threshold_	114
17.6.4.3 order_accuracy_	114
17.7 mtk::GLPKAdapter Class Reference	115

17.7.1	Detailed Description	115
17.7.2	Member Function Documentation	115
17.7.2.1	SolveSimplexAndCompare	115
17.8	mtk::Grad1D Class Reference	117
17.8.1	Detailed Description	121
17.8.2	Constructor & Destructor Documentation	121
17.8.2.1	Grad1D	121
17.8.2.2	Grad1D	121
17.8.2.3	~Grad1D	121
17.8.3	Member Function Documentation	121
17.8.3.1	AssembleOperator	121
17.8.3.2	coeffs_interior	121
17.8.3.3	ComputePreliminaryApproximations	122
17.8.3.4	ComputeRationalBasisNullSpace	122
17.8.3.5	ComputeStencilBoundaryGrid	123
17.8.3.6	ComputeStencilInteriorGrid	123
17.8.3.7	ComputeWeights	124
17.8.3.8	ConstructGrad1D	124
17.8.3.9	mim_bndy	126
17.8.3.10	mimetic_measure	127
17.8.3.11	num_bndy_coeffs	127
17.8.3.12	num_feasible_sols	127
17.8.3.13	ReturnAsDenseMatrix	128
17.8.3.14	ReturnAsDenseMatrix	129
17.8.3.15	ReturnAsDimensionlessDenseMatrix	130
17.8.3.16	sums_rows_mim_bndy	131
17.8.3.17	weights_cbs	131
17.8.3.18	weights_crs	131
17.8.4	Friends And Related Function Documentation	131
17.8.4.1	operator<<	131
17.8.5	Member Data Documentation	131
17.8.5.1	coeffs_interior_	131
17.8.5.2	dim_null_	132
17.8.5.3	gradient_	132
17.8.5.4	gradient_length_	132
17.8.5.5	mim_bndy_	132
17.8.5.6	mimetic_measure_	132

17.8.5.7 mimetic_threshold_	132
17.8.5.8 minrow_	132
17.8.5.9 num_bndy_approxs_	132
17.8.5.10 num_bndy_coeffs_	132
17.8.5.11 num_feasible_sols_	132
17.8.5.12 order_accuracy_	132
17.8.5.13 prem_apps_	133
17.8.5.14 rat_basis_null_space_	133
17.8.5.15 row_	133
17.8.5.16 sums_rows_mim_bndy_	133
17.8.5.17 weights_cbs_	133
17.8.5.18 weights_crs_	133
17.9 mtk::Grad2D Class Reference	133
17.9.1 Detailed Description	135
17.9.2 Constructor & Destructor Documentation	135
17.9.2.1 Grad2D	135
17.9.2.2 ~Grad2D	135
17.9.2.3 ~Grad2D	135
17.9.3 Member Function Documentation	136
17.9.3.1 ConstructGrad2D	136
17.9.3.2 ReturnAsDenseMatrix	136
17.9.4 Member Data Documentation	137
17.9.4.1 gradient_	137
17.9.4.2 mimetic_threshold_	137
17.9.4.3 order_accuracy_	137
17.10 mtk::Grad3D Class Reference	137
17.10.1 Detailed Description	139
17.10.2 Constructor & Destructor Documentation	139
17.10.2.1 Grad3D	139
17.10.2.2 ~Grad3D	139
17.10.2.3 ~Grad3D	139
17.10.3 Member Function Documentation	140
17.10.3.1 ConstructGrad3D	140
17.10.3.2 ReturnAsDenseMatrix	141
17.10.4 Member Data Documentation	141
17.10.4.1 gradient_	141
17.10.4.2 mimetic_threshold_	141

17.10.4.3 <code>order_accuracy_</code>	141
17.11 <code>mtk::Interp1D</code> Class Reference	142
17.11.1 Detailed Description	143
17.11.2 Constructor & Destructor Documentation	143
17.11.2.1 <code>Interp1D</code>	143
17.11.2.2 <code>Interp1D</code>	143
17.11.2.3 <code>~Interp1D</code>	143
17.11.3 Member Function Documentation	143
17.11.3.1 <code>coeffs_interior_</code>	143
17.11.3.2 <code>ConstructInterp1D</code>	143
17.11.3.3 <code>ReturnAsDenseMatrix</code>	144
17.11.4 Friends And Related Function Documentation	145
17.11.4.1 <code>operator<<</code>	145
17.11.5 Member Data Documentation	145
17.11.5.1 <code>coeffs_interior_</code>	145
17.11.5.2 <code>dir_interp_</code>	145
17.11.5.3 <code>order_accuracy_</code>	145
17.12 <code>mtk::Interp2D</code> Class Reference	145
17.12.1 Detailed Description	147
17.12.2 Constructor & Destructor Documentation	147
17.12.2.1 <code>Interp2D</code>	147
17.12.2.2 <code>Interp2D</code>	147
17.12.2.3 <code>~Interp2D</code>	147
17.12.3 Member Function Documentation	147
17.12.3.1 <code>ConstructInterp2D</code>	147
17.12.3.2 <code>ReturnAsDenseMatrix</code>	148
17.12.4 Member Data Documentation	148
17.12.4.1 <code>interpolator_</code>	148
17.12.4.2 <code>mimetic_threshold_</code>	148
17.12.4.3 <code>order_accuracy_</code>	148
17.13 <code>mtk::Lap1D</code> Class Reference	148
17.13.1 Detailed Description	151
17.13.2 Constructor & Destructor Documentation	151
17.13.2.1 <code>Lap1D</code>	151
17.13.2.2 <code>Lap1D</code>	151
17.13.2.3 <code>~Lap1D</code>	151
17.13.3 Member Function Documentation	151

17.13.3.1 ConstructLap1D	151
17.13.3.2 data	153
17.13.3.3 delta	153
17.13.3.4 mimetic_measure	154
17.13.3.5 mimetic_threshold	154
17.13.3.6 order_accuracy	154
17.13.3.7 ReturnAsDenseMatrix	155
17.13.3.8 sums_rows_mim_bndy	156
17.13.4 Friends And Related Function Documentation	157
17.13.4.1 operator<<	157
17.13.5 Member Data Documentation	157
17.13.5.1 delta_	157
17.13.5.2 laplacian_	157
17.13.5.3 laplacian_length_	157
17.13.5.4 mimetic_measure_	157
17.13.5.5 mimetic_threshold_	157
17.13.5.6 order_accuracy_	157
17.13.5.7 sums_rows_mim_bndy_	157
17.14 mtk::Lap2D Class Reference	158
17.14.1 Detailed Description	159
17.14.2 Constructor & Destructor Documentation	159
17.14.2.1 Lap2D	159
17.14.2.2 Lap2D	159
17.14.2.3 ~Lap2D	159
17.14.3 Member Function Documentation	160
17.14.3.1 ConstructLap2D	160
17.14.3.2 data	161
17.14.3.3 ReturnAsDenseMatrix	161
17.14.4 Member Data Documentation	161
17.14.4.1 laplacian_	161
17.14.4.2 mimetic_threshold_	162
17.14.4.3 order_accuracy_	162
17.15 mtk::Lap3D Class Reference	162
17.15.1 Detailed Description	164
17.15.2 Constructor & Destructor Documentation	164
17.15.2.1 Lap3D	164
17.15.2.2 Lap3D	164

17.15.2.3 ~Lap3D	164
17.15.3 Member Function Documentation	165
17.15.3.1 ConstructLap3D	165
17.15.3.2 data	166
17.15.3.3 operator*	166
17.15.3.4 ReturnAsDenseMatrix	166
17.15.4 Member Data Documentation	166
17.15.4.1 laplacian_	166
17.15.4.2 mimetic_threshold_	166
17.15.4.3 order_accuracy_	167
17.16 mtk::LAPACKAdapter Class Reference	167
17.16.1 Detailed Description	168
17.16.2 Member Function Documentation	168
17.16.2.1 QRFactorDenseMatrix	168
17.16.2.2 SolveDenseSystem	169
17.16.2.3 SolveDenseSystem	170
17.16.2.4 SolveDenseSystem	171
17.16.2.5 SolveDenseSystem	172
17.16.2.6 SolveRectangularDenseSystem	173
17.17 mtk::Matrix Class Reference	174
17.17.1 Detailed Description	178
17.17.2 Constructor & Destructor Documentation	178
17.17.2.1 Matrix	178
17.17.2.2 Matrix	178
17.17.2.3 ~Matrix	179
17.17.3 Member Function Documentation	179
17.17.3.1 abs_density	179
17.17.3.2 abs_sparsity	179
17.17.3.3 bandwidth	180
17.17.3.4 ComputeAbsDensity	180
17.17.3.5 ComputeAbsSparsity	181
17.17.3.6 ComputeBandwidth	181
17.17.3.7 ComputeLeadingDimension	182
17.17.3.8 ComputeNumNonNull	182
17.17.3.9 ComputeNumNonZero	183
17.17.3.10 ComputeNumValues	183
17.17.3.11 ComputeRelDensity	184

17.17.3.12	ComputeRelSparsity	185
17.17.3.13	DecreaseNumNull	185
17.17.3.14	DecreaseNumZero	186
17.17.3.15	encoded_operator	186
17.17.3.16	IncreaseNumNull	187
17.17.3.17	IncreaseNumZero	188
17.17.3.18	leading_dimension	188
17.17.3.19	num_cols	189
17.17.3.20	num_low_diags	189
17.17.3.21	num_non_null	189
17.17.3.22	num_non_zero	190
17.17.3.23	num_null	190
17.17.3.24	num_rows	191
17.17.3.25	num_upp_diags	192
17.17.3.26	num_values	192
17.17.3.27	num_zero	192
17.17.3.28	ordering	193
17.17.3.29	el_density	194
17.17.3.30	el_sparsity	195
17.17.3.31	set_encoded_operator	195
17.17.3.32	set_num_cols	196
17.17.3.33	set_num_low_diags	197
17.17.3.34	set_num_null	198
17.17.3.35	set_num_rows	198
17.17.3.36	set_num_upp_diags	199
17.17.3.37	set_num_zero	200
17.17.3.38	set_ordering	200
17.17.3.39	set_storage	201
17.17.3.40	storage	202
17.17.4	Member Data Documentation	203
17.17.4.1	abs_density_	203
17.17.4.2	abs_sparsity_	203
17.17.4.3	bandwidth_	203
17.17.4.4	encoded_operator_	203
17.17.4.5	leading_dimension_	203
17.17.4.6	num_cols_	204
17.17.4.7	num_low_diags_	204

17.17.4.8 num_non_null_	204
17.17.4.9 num_non_zero_	204
17.17.4.10 num_null_	204
17.17.4.11 num_rows_	204
17.17.4.12 num_upp_diags_	204
17.17.4.13 num_values_	204
17.17.4.14 num_zero_	204
17.17.4.15 ordering_	204
17.17.4.16 el_density_	204
17.17.4.17 rel_sparsity_	205
17.17.4.18 storage_	205
17.18 mtk::OperatorApplicator Class Reference	205
17.18.1 Detailed Description	206
17.18.2 Member Function Documentation	206
17.18.2.1 ApplyDenseMatrixDivergenceOn1DGrid	206
17.18.2.2 ApplyDenseMatrixGradientOn1DGrid	206
17.18.2.3 ApplyDenseMatrixLaplacianOn1DGrid	207
17.19 mtk::Quad1D Class Reference	208
17.19.1 Detailed Description	209
17.19.2 Constructor & Destructor Documentation	209
17.19.2.1 Quad1D	209
17.19.2.2 Quad1D	209
17.19.2.3 ~Quad1D	210
17.19.3 Member Function Documentation	210
17.19.3.1 degree_approximation	210
17.19.3.2 Integrate	210
17.19.3.3 weights	210
17.19.4 Friends And Related Function Documentation	210
17.19.4.1 operator<<	210
17.19.5 Member Data Documentation	210
17.19.5.1 degree_approximation_	210
17.19.5.2 weights_	210
17.20 mtk::RobinBCDescriptor1D Class Reference	210
17.20.1 Detailed Description	213
17.20.2 Constructor & Destructor Documentation	213
17.20.2.1 RobinBCDescriptor1D	213
17.20.2.2 RobinBCDescriptor1D	213

17.20.2.3 ~RobinBCDescriptor1D	213
17.20.3 Member Function Documentation	213
17.20.3.1 highest_order_diff_east	213
17.20.3.2 highest_order_diff_west	214
17.20.3.3 ImposeOnDivergenceMatrix	214
17.20.3.4 ImposeOnGrid	214
17.20.3.5 ImposeOnLaplacianMatrix	215
17.20.3.6 PushBackEastCoeff	217
17.20.3.7 PushBackWestCoeff	217
17.20.3.8 set_east_condition	218
17.20.3.9 set_west_condition	219
17.20.4 Member Data Documentation	219
17.20.4.1 east_coefficients_	219
17.20.4.2 east_condition_	220
17.20.4.3 highest_order_diff_east_	220
17.20.4.4 highest_order_diff_west_	220
17.20.4.5 west_coefficients_	220
17.20.4.6 west_condition_	220
17.21 mtk::RobinBCDescriptor2D Class Reference	220
17.21.1 Detailed Description	224
17.21.2 Constructor & Destructor Documentation	224
17.21.2.1 RobinBCDescriptor2D	224
17.21.2.2 RobinBCDescriptor2D	224
17.21.2.3 ~RobinBCDescriptor2D	224
17.21.3 Member Function Documentation	224
17.21.3.1 highest_order_diff_east	224
17.21.3.2 highest_order_diff_north	225
17.21.3.3 highest_order_diff_south	225
17.21.3.4 highest_order_diff_west	226
17.21.3.5 ImposeOnEastBoundaryNoSpace	226
17.21.3.6 ImposeOnEastBoundaryWithSpace	227
17.21.3.7 ImposeOnGrid	228
17.21.3.8 ImposeOnLaplacianMatrix	231
17.21.3.9 ImposeOnNorthBoundaryNoSpace	232
17.21.3.10 ImposeOnNorthBoundaryWithSpace	233
17.21.3.11 ImposeOnSouthBoundaryNoSpace	234
17.21.3.12 ImposeOnSouthBoundaryWithSpace	235

17.21.3.13	ImposeOnWestBoundaryNoSpace	236
17.21.3.14	ImposeOnWestBoundaryWithSpace	237
17.21.3.15	PushBackEastCoeff	238
17.21.3.16	PushBackNorthCoeff	238
17.21.3.17	PushBackSouthCoeff	239
17.21.3.18	PushBackWestCoeff	240
17.21.3.19	set_east_condition	240
17.21.3.20	set_north_condition	241
17.21.3.21	set_south_condition	242
17.21.3.22	set_west_condition	242
17.21.4	Member Data Documentation	243
17.21.4.1	east_coefficients_	243
17.21.4.2	east_condition_	243
17.21.4.3	highest_order_diff_east_	243
17.21.4.4	highest_order_diff_north_	243
17.21.4.5	highest_order_diff_south_	243
17.21.4.6	highest_order_diff_west_	244
17.21.4.7	north_coefficients_	244
17.21.4.8	north_condition_	244
17.21.4.9	south_coefficients_	244
17.21.4.10	south_condition_	244
17.21.4.11	west_coefficients_	244
17.21.4.12	west_condition_	244
17.22	mtk::RobinBCDescriptor3D Class Reference	244
17.22.1	Detailed Description	248
17.22.2	Constructor & Destructor Documentation	248
17.22.2.1	RobinBCDescriptor3D	248
17.22.2.2	RobinBCDescriptor3D	248
17.22.2.3	~RobinBCDescriptor3D	248
17.22.3	Member Function Documentation	248
17.22.3.1	highest_order_diff_west	248
17.22.3.2	ImposeOnEastBoundaryNoSpace	248
17.22.3.3	ImposeOnEastBoundaryWithSpace	249
17.22.3.4	ImposeOnGrid	249
17.22.3.5	ImposeOnLaplacianMatrix	249
17.22.3.6	ImposeOnNorthBoundaryNoSpace	249
17.22.3.7	ImposeOnNorthBoundaryWithSpace	249

17.22.3.8 ImposeOnSouthBoundaryNoSpace	250
17.22.3.9 ImposeOnSouthBoundaryWithSpace	250
17.22.3.10ImposeOnWestBoundaryNoSpace	250
17.22.3.11ImposeOnWestBoundaryWithSpace	250
17.22.3.12PushBackWestCoeff	251
17.22.3.13set_west_condition	251
17.22.4 Member Data Documentation	251
17.22.4.1 bottom_coefficients_	251
17.22.4.2 bottom_condition_	251
17.22.4.3 east_coefficients_	251
17.22.4.4 east_condition_	251
17.22.4.5 highest_order_diff_bottom_	251
17.22.4.6 highest_order_diff_east_	251
17.22.4.7 highest_order_diff_north_	251
17.22.4.8 highest_order_diff_south_	252
17.22.4.9 highest_order_diff_top_	252
17.22.4.10highest_order_diff_west_	252
17.22.4.11north_coefficients_	252
17.22.4.12north_condition_	252
17.22.4.13south_coefficients_	252
17.22.4.14south_condition_	252
17.22.4.15top_coefficients_	252
17.22.4.16top_condition_	252
17.22.4.17west_coefficients_	252
17.22.4.18west_condition_	253
17.23mtk::Tools Class Reference	253
17.23.1 Detailed Description	254
17.23.2 Member Function Documentation	254
17.23.2.1 Assert	254
17.23.2.2 BeginUnitTestNo	255
17.23.2.3 EndUnitTestNo	257
17.23.2.4 Prevent	259
17.23.3 Member Data Documentation	260
17.23.3.1 begin_time_	260
17.23.3.2 duration_	260
17.23.3.3 test_number_	260
17.24mtk::UniStgGrid1D Class Reference	260

17.24.1 Detailed Description	263
17.24.2 Constructor & Destructor Documentation	263
17.24.2.1 UniStgGrid1D	263
17.24.2.2 UniStgGrid1D	263
17.24.2.3 UniStgGrid1D	263
17.24.2.4 ~UniStgGrid1D	264
17.24.3 Member Function Documentation	264
17.24.3.1 BindScalarField	264
17.24.3.2 BindScalarField	265
17.24.3.3 BindVectorField	265
17.24.3.4 delta_x	266
17.24.3.5 discrete_domain_x	267
17.24.3.6 discrete_field	267
17.24.3.7 east_bndy_x	268
17.24.3.8 field_nature	268
17.24.3.9 GenerateDiscreteDomainX	269
17.24.3.10 num_cells_x	269
17.24.3.11 operator=	270
17.24.3.12 ReserveDiscreteField	271
17.24.3.13 west_bndy_x	271
17.24.3.14 WriteToFile	271
17.24.4 Friends And Related Function Documentation	272
17.24.4.1 operator<<	272
17.24.5 Member Data Documentation	272
17.24.5.1 delta_x_	272
17.24.5.2 discrete_domain_x_	272
17.24.5.3 discrete_field_	272
17.24.5.4 east_bndy_x_	272
17.24.5.5 field_nature_	272
17.24.5.6 num_cells_x_	273
17.24.5.7 west_bndy_x_	273
17.25 mtk::UniStgGrid2D Class Reference	273
17.25.1 Detailed Description	276
17.25.2 Constructor & Destructor Documentation	277
17.25.2.1 UniStgGrid2D	277
17.25.2.2 UniStgGrid2D	277
17.25.2.3 UniStgGrid2D	277

17.25.2.4 ~UniStgGrid2D	277
17.25.3 Member Function Documentation	278
17.25.3.1 BindScalarField	278
17.25.3.2 BindVectorField	278
17.25.3.3 BindVectorFieldPComponent	279
17.25.3.4 BindVectorFieldQComponent	280
17.25.3.5 Bound	280
17.25.3.6 delta_x	280
17.25.3.7 delta_y	281
17.25.3.8 discrete_domain_x	281
17.25.3.9 discrete_domain_y	282
17.25.3.10 discrete_field	282
17.25.3.11 east_bndy	283
17.25.3.12 nature	283
17.25.3.13 north_bndy	284
17.25.3.14 num_cells_x	284
17.25.3.15 num_cells_y	285
17.25.3.16 size	286
17.25.3.17 south_bndy	286
17.25.3.18 west_bndy	287
17.25.3.19 WriteToFile	287
17.25.4 Friends And Related Function Documentation	288
17.25.4.1 operator<<	288
17.25.5 Member Data Documentation	288
17.25.5.1 delta_x_	288
17.25.5.2 delta_y_	289
17.25.5.3 discrete_domain_x_	289
17.25.5.4 discrete_domain_y_	289
17.25.5.5 discrete_field_	289
17.25.5.6 east_bndy_	289
17.25.5.7 nature_	289
17.25.5.8 north_bndy_	289
17.25.5.9 num_cells_x_	289
17.25.5.10 num_cells_y_	289
17.25.5.11 south_bndy_	289
17.25.5.12 west_bndy_	289
17.26 mtk::UniStgGrid3D Class Reference	290

17.26.1 Detailed Description	294
17.26.2 Constructor & Destructor Documentation	294
17.26.2.1 UniStgGrid3D	294
17.26.2.2 UniStgGrid3D	294
17.26.2.3 UniStgGrid3D	294
17.26.2.4 ~UniStgGrid3D	295
17.26.3 Member Function Documentation	295
17.26.3.1 BindScalarField	295
17.26.3.2 BindVectorField	296
17.26.3.3 BindVectorFieldPComponent	297
17.26.3.4 BindVectorFieldQComponent	297
17.26.3.5 BindVectorFieldRComponent	297
17.26.3.6 bottom_bndy	298
17.26.3.7 Bound	298
17.26.3.8 delta_x	298
17.26.3.9 delta_y	299
17.26.3.10delta_z	299
17.26.3.11discrete_domain_x	300
17.26.3.12discrete_domain_y	300
17.26.3.13discrete_domain_z	300
17.26.3.14discrete_field	300
17.26.3.15east_bndy	301
17.26.3.16nature	301
17.26.3.17north_bndy	301
17.26.3.18num_cells_x	302
17.26.3.19num_cells_y	302
17.26.3.20num_cells_z	302
17.26.3.21operator=	303
17.26.3.22Size	303
17.26.3.23south_bndy	303
17.26.3.24top_bndy	304
17.26.3.25west_bndy	304
17.26.3.26WriteToFile	305
17.26.4 Friends And Related Function Documentation	306
17.26.4.1 operator<<	306
17.26.5 Member Data Documentation	306
17.26.5.1 bottom_bndy_	306

17.26.5.2 delta_x_	306
17.26.5.3 delta_y_	306
17.26.5.4 delta_z_	306
17.26.5.5 discrete_domain_x_	306
17.26.5.6 discrete_domain_y_	306
17.26.5.7 discrete_domain_z_	306
17.26.5.8 discrete_field_	306
17.26.5.9 east_bndy_	306
17.26.5.10 nature_	307
17.26.5.11 north_bndy_	307
17.26.5.12 num_cells_x_	307
17.26.5.13 num_cells_y_	307
17.26.5.14 num_cells_z_	307
17.26.5.15 south_bndy_	307
17.26.5.16 op_bndy_	307
17.26.5.17 west_bndy_	307
18 File Documentation	309
18.1 examples/1d_accuracy/1d_accuracy.cc File Reference	309
18.1.1 Detailed Description	309
18.1.2 Function Documentation	310
18.1.2.1 main	310
18.2 1d_accuracy.cc	310
18.3 examples/1d_divergence/1d_divergence.cc File Reference	314
18.3.1 Detailed Description	314
18.3.2 Function Documentation	314
18.3.2.1 main	314
18.4 1d_divergence.cc	314
18.5 examples/1d_gradient/1d_gradient.cc File Reference	316
18.5.1 Detailed Description	316
18.5.2 Function Documentation	316
18.5.2.1 main	317
18.6 1d_gradient.cc	317
18.7 examples/1d_laplacian/1d_laplacian.cc File Reference	318
18.7.1 Detailed Description	318
18.7.2 Function Documentation	319
18.7.2.1 main	319

18.8 1d_laplacian.cc	319
18.9 examples/1d_mimetic_measure/1d_mimetic_measure.cc File Reference	320
18.9.1 Detailed Description	321
18.9.2 Function Documentation	321
18.9.2.1 main	321
18.10 1d_mimetic_measure.cc	321
18.11 examples/1d_mimetic_threshold/1d_mimetic_threshold.cc File Reference	322
18.11.1 Detailed Description	323
18.11.2 Function Documentation	323
18.11.2.1 main	323
18.12 1d_mimetic_threshold.cc	323
18.13 examples/1d_poisson/1d_poisson.cc File Reference	325
18.13.1 Detailed Description	326
18.13.2 Function Documentation	327
18.13.2.1 Alpha	327
18.13.2.2 Beta	327
18.13.2.3 Epsilon	327
18.13.2.4 KnownSolution	328
18.13.2.5 main	328
18.13.2.6 Omega	329
18.13.2.7 Source	330
18.14 1d_poisson.cc	330
18.15 examples/1d_poisson_minimal/1d_poisson_minimal.cc File Reference	333
18.15.1 Detailed Description	333
18.15.2 Function Documentation	334
18.15.2.1 Alpha	334
18.15.2.2 Beta	334
18.15.2.3 Epsilon	335
18.15.2.4 KnownSolution	335
18.15.2.5 main	335
18.15.2.6 Omega	336
18.15.2.7 Source	337
18.16 1d_poisson_minimal.cc	337
18.17 examples/1d_poisson_sensitivity_threshold/1d_poisson_sensitivity_threshold.cc File Reference	339
18.17.1 Detailed Description	339
18.17.2 Function Documentation	340
18.17.2.1 Alpha	340

18.17.2.2 Beta	340
18.17.2.3 Epsilon	341
18.17.2.4 KnownSolution	341
18.17.2.5 main	341
18.17.2.6 Omega	342
18.17.2.7 Source	343
18.181d_poisson_sensitivity_threshold.cc	343
18.19examples/1d_poisson_supercritical/1d_poisson_supercritical.cc File Reference	347
18.19.1 Detailed Description	347
18.19.2 Function Documentation	348
18.19.2.1 Alpha	348
18.19.2.2 Beta	348
18.19.2.3 Epsilon	348
18.19.2.4 KnownSolution	349
18.19.2.5 main	349
18.19.2.6 Omega	350
18.19.2.7 Source	351
18.201d_poisson_supercritical.cc	351
18.21examples/1d_positive_weights/1d_positive_weights.cc File Reference	354
18.21.1 Detailed Description	355
18.21.2 Function Documentation	355
18.21.2.1 main	355
18.221d_positive_weights.cc	355
18.23examples/2d_angular_velocity/2d_angular_velocity.cc File Reference	357
18.23.1 Detailed Description	358
18.23.2 Function Documentation	358
18.23.2.1 main	358
18.242d_angular_velocity.cc	358
18.25examples/2d_poisson/2d_poisson.cc File Reference	359
18.25.1 Detailed Description	360
18.25.2 Function Documentation	361
18.25.2.1 main	361
18.262d_poisson.cc	361
18.27examples/3d_diffusion/3d_diffusion.cc File Reference	363
18.27.1 Detailed Description	364
18.27.2 Function Documentation	364
18.27.2.1 main	364

18.283d_diffusion.cc	364
18.29include/mtk.h File Reference	366
18.29.1 Detailed Description	367
18.30mtk.h	368
18.31include/mtk blas_adapter.h File Reference	369
18.31.1 Detailed Description	370
18.32mtk blas_adapter.h	370
18.33include/mtk_curl_2d.h File Reference	371
18.33.1 Detailed Description	372
18.34mtk_curl_2d.h	372
18.35include/mtk_dense_matrix.h File Reference	373
18.35.1 Detailed Description	374
18.36mtk_dense_matrix.h	375
18.37include/mtk_div_1d.h File Reference	376
18.37.1 Detailed Description	377
18.38mtk_div_1d.h	377
18.39include/mtk_div_2d.h File Reference	379
18.39.1 Detailed Description	380
18.40mtk_div_2d.h	380
18.41include/mtk_div_3d.h File Reference	381
18.41.1 Detailed Description	382
18.42mtk_div_3d.h	382
18.43include/mtkEnums.h File Reference	383
18.43.1 Detailed Description	384
18.44mtkEnums.h	384
18.45include/mtk_foundations.h File Reference	385
18.45.1 Detailed Description	386
18.46mtk_foundations.h	386
18.47include/mtk_glpk_adapter.h File Reference	387
18.47.1 Detailed Description	388
18.48mtk_glpk_adapter.h	389
18.49include/mtk_grad_1d.h File Reference	390
18.49.1 Detailed Description	390
18.50mtk_grad_1d.h	391
18.51include/mtk_grad_2d.h File Reference	392
18.51.1 Detailed Description	393
18.52mtk_grad_2d.h	393

18.53include/mtk_grad_3d.h File Reference	394
18.53.1 Detailed Description	395
18.54mtk_grad_3d.h	395
18.55include/mtk_interp_1d.h File Reference	397
18.55.1 Detailed Description	397
18.56mtk_interp_1d.h	398
18.57include/mtk_interp_2d.h File Reference	399
18.57.1 Detailed Description	399
18.58mtk_interp_2d.h	400
18.59include/mtk_lap_1d.h File Reference	401
18.59.1 Detailed Description	401
18.60mtk_lap_1d.h	402
18.61include/mtk_lap_2d.h File Reference	403
18.61.1 Detailed Description	404
18.62mtk_lap_2d.h	404
18.63include/mtk_lap_3d.h File Reference	405
18.63.1 Detailed Description	406
18.64mtk_lap_3d.h	406
18.65include/mtk_lapack_adapter.h File Reference	407
18.65.1 Detailed Description	408
18.66mtk_lapack_adapter.h	408
18.67include/mtk_matrix.h File Reference	409
18.67.1 Detailed Description	410
18.68mtk_matrix.h	410
18.69include/mtk_operator_applicator.h File Reference	413
18.69.1 Detailed Description	414
18.70mtk_operator_applicator.h	414
18.71include/mtk_quad_1d.h File Reference	415
18.71.1 Detailed Description	416
18.72mtk_quad_1d.h	416
18.73include/mtk_robin_bc_descriptor_1d.h File Reference	417
18.73.1 Detailed Description	418
18.74mtk_robin_bc_descriptor_1d.h	419
18.75include/mtk_robin_bc_descriptor_2d.h File Reference	420
18.75.1 Detailed Description	421
18.76mtk_robin_bc_descriptor_2d.h	422
18.77include/mtk_robin_bc_descriptor_3d.h File Reference	424

18.77.1 Detailed Description	425
18.78mtk_robin_bc_descriptor_3d.h	425
18.79include/mtk_tools.h File Reference	427
18.79.1 Detailed Description	428
18.80mtk_tools.h	428
18.81include/mtk_uni_stg_grid_1d.h File Reference	429
18.81.1 Detailed Description	430
18.82mtk_uni_stg_grid_1d.h	431
18.83include/mtk_uni_stg_grid_2d.h File Reference	432
18.83.1 Detailed Description	433
18.84mtk_uni_stg_grid_2d.h	433
18.85include/mtk_uni_stg_grid_3d.h File Reference	435
18.85.1 Detailed Description	436
18.86mtk_uni_stg_grid_3d.h	436
18.87README.md File Reference	439
18.88README.md	439
18.89src/mtk blas_adapter.cc File Reference	441
18.89.1 Detailed Description	441
18.90mtk blas_adapter.cc	442
18.91src/mtk_curl_2d.cc File Reference	447
18.91.1 Detailed Description	447
18.92mtk_curl_2d.cc	447
18.93src/mtk_dense_matrix.cc File Reference	449
18.94mtk_dense_matrix.cc	450
18.95src/mtk_div_1d.cc File Reference	458
18.95.1 Detailed Description	458
18.96mtk_div_1d.cc	459
18.97src/mtk_div_2d.cc File Reference	477
18.97.1 Detailed Description	478
18.98mtk_div_2d.cc	478
18.99src/mtk_div_3d.cc File Reference	480
18.99.1 Detailed Description	481
18.100mtk_div_3d.cc	481
18.101src/mtk_glpk_adapter.cc File Reference	483
18.101.1 Detailed Description	484
18.102mtk_glpk_adapter.cc	484
18.103rc/mtk_grad_1d.cc File Reference	488

18.103. Detailed Description	489
18.104 mtk_grad_1d.cc	489
18.105 rc/mtk_grad_2d.cc File Reference	509
18.105. Detailed Description	510
18.106 mtk_grad_2d.cc	510
18.107 rc/mtk_grad_3d.cc File Reference	512
18.107. Detailed Description	512
18.108 mtk_grad_3d.cc	512
18.109 rc/mtk_interp_1d.cc File Reference	514
18.109. Detailed Description	515
18.110 mtk_interp_1d.cc	515
18.111 rc/mtk_lap_1d.cc File Reference	518
18.111. Detailed Description	518
18.112 mtk_lap_1d.cc	518
18.113 rc/mtk_lap_2d.cc File Reference	523
18.113. Detailed Description	524
18.114 mtk_lap_2d.cc	524
18.115 rc/mtk_lap_3d.cc File Reference	525
18.115. Detailed Description	526
18.116 mtk_lap_3d.cc	526
18.117 rc/mtk_lapack_adapter.cc File Reference	528
18.117. Detailed Description	529
18.118 mtk_lapack_adapter.cc	529
18.119 rc/mtk_matrix.cc File Reference	537
18.119. Detailed Description	537
18.120 mtk_matrix.cc	537
18.121 rc/mtk_operator_applicator.cc File Reference	544
18.121. Detailed Description	544
18.122 mtk_operator_applicator.cc	544
18.123 rc/mtk_robin_bc_descriptor_1d.cc File Reference	546
18.123. Detailed Description	546
18.124 mtk_robin_bc_descriptor_1d.cc	547
18.125 rc/mtk_robin_bc_descriptor_2d.cc File Reference	550
18.125. Detailed Description	550
18.126 mtk_robin_bc_descriptor_2d.cc	551
18.127 rc/mtk_tools.cc File Reference	560
18.127. Detailed Description	561

18.128	ntk_tools.cc	561
18.129	rc/mtk_uni_stg_grid_1d.cc File Reference	562
18.129.1	Detailed Description	563
18.130	ntk_uni_stg_grid_1d.cc	563
18.131	rc/mtk_uni_stg_grid_2d.cc File Reference	568
18.131.1	Detailed Description	568
18.132	ntk_uni_stg_grid_2d.cc	569
18.133	rc/mtk_uni_stg_grid_3d.cc File Reference	575
18.133.1	Detailed Description	576
18.134	ntk_uni_stg_grid_3d.cc	576
18.135	tests/mtk blas adapter test/mtk blas adapter test.cc File Reference	582
18.135.1	Detailed Description	582
18.135.2	Function Documentation	582
18.135.2.1	main	582
18.135.2.2	TestRealDenseMM	583
18.136	ntk blas adapter test.cc	584
18.137	tests/mtk dense matrix test/mtk dense matrix test.cc File Reference	585
18.137.1	Detailed Description	586
18.137.2	Function Documentation	586
18.137.2.1	main	586
18.137.2.2	TestConstructAsIdentity	586
18.137.2.3	TestConstructAsVandermonde	587
18.137.2.4	TestConstructAsVandermondeTranspose	588
18.137.2.5	TestConstructAsVandermondeTransposeAssignmentOperator	588
18.137.2.6	TestConstructorWithNumRowsNumCols	589
18.137.2.7	TestConstructWithNumRowsNumColsAssignmentOperator	590
18.137.2.8	TestDefaultConstructor	590
18.137.2.9	TestKron	591
18.137.2.10	TestSetValueGetValue	592
18.138	ntk dense matrix test.cc	593
18.139	tests/mtk div 1d test/mtk div 1d test.cc File Reference	597
18.139.1	Detailed Description	597
18.139.2	Function Documentation	597
18.139.2.1	main	597
18.139.2.2	TestDefaultConstructorFactoryMethodDefault	598
18.139.2.3	TestDefaultConstructorFactoryMethodEightOrderDefThreshold	599
18.139.2.4	TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold	599

18.139.2.5TestDefaultConstructorFactoryMethodFourthOrder	600
18.139.2.6TestDefaultConstructorFactoryMethodSixthOrder	601
18.139.2.7TestDefaultConstructorFactoryMethodTenthOrderDefThreshold	601
18.139.2.8TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold	602
18.139.2.9TestFourthOrderReturnAsDenseMatrixWithGrid	603
18.139.2.10TestSecondOrderReturnAsDenseMatrixWithGrid	603
18.139.2.11TestSixthOrderReturnAsDenseMatrixWithGrid	604
18.140mtk_div_1d_test.cc	605
18.141tests/mtk_div_2d_test/mtk_div_2d_test.cc File Reference	609
18.141.1Detailed Description	609
18.141.2Function Documentation	610
18.141.2.1main	610
18.141.2.2TestDefaultConstructorFactory	610
18.141.2.3TestReturnAsDenseMatrixWriteToFile	611
18.142mtk_div_2d_test.cc	612
18.143tests/mtk_div_3d_test/mtk_div_3d_test.cc File Reference	614
18.143.1Detailed Description	614
18.143.2Function Documentation	615
18.143.2.1main	615
18.143.2.2TestDefaultConstructorFactory	615
18.143.2.3TestReturnAsDenseMatrixWriteToFile	616
18.144mtk_div_3d_test.cc	617
18.145tests/mtk_glpk_adapter_test/mtk_glpk_adapter_test.cc File Reference	619
18.145.1Detailed Description	619
18.145.2Function Documentation	620
18.145.2.1main	620
18.145.2.2Test1	620
18.146mtk_glpk_adapter_test.cc	621
18.147tests/mtk_grad_1d_test/mtk_grad_1d_test.cc File Reference	621
18.147.1Detailed Description	622
18.147.2Function Documentation	623
18.147.2.1main	623
18.147.2.2TestDefaultConstructorFactoryMethodDefault	623
18.147.2.3TestDefaultConstructorFactoryMethodEightOrderDefThreshold	624
18.147.2.4TestDefaultConstructorFactoryMethodFourthOrder	624
18.147.2.5TestDefaultConstructorFactoryMethodSixthOrder	625
18.147.2.6TestDefaultConstructorFactoryMethodTenthOrderDefThreshold	626

18.147.2.7TestMimBndy	626
18.147.2.8TestReturnAsDenseMatrixWithGrid	627
18.147.2.9TestReturnAsDimensionlessDenseMatrix	628
18.147.2.10testWriteToFile	628
18.148mtk_grad_1d_test.cc	629
18.149tests/mtk_grad_2d_test/mtk_grad_2d_test.cc File Reference	633
18.149.1Detailed Description	634
18.149.2Function Documentation	634
18.149.2.1main	634
18.149.2.2TestDefaultConstructorFactory	634
18.149.2.3TestReturnAsDenseMatrixWriteToFile	635
18.150mtk_grad_2d_test.cc	636
18.151tests/mtk_grad_3d_test/mtk_grad_3d_test.cc File Reference	638
18.151.1Detailed Description	638
18.151.2Function Documentation	638
18.151.2.1main	639
18.151.2.2TestDefaultConstructorFactory	639
18.151.2.3TestReturnAsDenseMatrixWriteToFile	640
18.152mtk_grad_3d_test.cc	641
18.153tests/mtk_interp_1d_test/mtk_interp_1d_test.cc File Reference	643
18.153.1Detailed Description	644
18.153.2Function Documentation	644
18.153.2.1main	644
18.153.2.2TestDefaultConstructorFactoryMethodDefault	644
18.153.2.3TestReturnAsDenseMatrixWithGrid	645
18.154mtk_interp_1d_test.cc	645
18.155tests/mtk_lap_1d_test/mtk_lap_1d_test.cc File Reference	647
18.155.1Detailed Description	647
18.155.2Function Documentation	647
18.155.2.1main	647
18.155.2.2TestDefaultConstructorFactoryMethodDefault	648
18.155.2.3TestDefaultConstructorFactoryMethodEightOrderDefThreshold	649
18.155.2.4TestDefaultConstructorFactoryMethodFourthOrder	650
18.155.2.5TestDefaultConstructorFactoryMethodSixthOrder	651
18.155.2.6TestDefaultConstructorFactoryMethodTenthOrderDefThreshold	652
18.155.2.7TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold	653
18.155.2.8TestReturnAsDenseMatrix	654

18.156 <code>mtk_lap_1d_test.cc</code>	655
18.157 <code>tests/mtk_lap_2d_test/mtk_lap_2d_test.cc</code> File Reference	658
18.157.1Detailed Description	658
18.157.2Function Documentation	658
18.157.2.1 <code>main</code>	658
18.157.2.2 <code>TestDefaultConstructorFactory</code>	659
18.157.2.3 <code>TestReturnAsDenseMatrixWriteToFile</code>	660
18.158 <code>mtk_lap_2d_test.cc</code>	661
18.159 <code>tests/mtk_lap_3d_test/mtk_lap_3d_test.cc</code> File Reference	663
18.159.1Detailed Description	663
18.159.2Function Documentation	664
18.159.2.1 <code>main</code>	664
18.159.2.2 <code>TestDefaultConstructorFactory</code>	664
18.159.2.3 <code>TestReturnAsDenseMatrixWriteToFile</code>	665
18.160 <code>mtk_lap_3d_test.cc</code>	666
18.161 <code>tests/mtk_lapack_adapter_test/mtk_lapack_adapter_test.cc</code> File Reference	668
18.161.1Detailed Description	669
18.161.2Function Documentation	669
18.161.2.1 <code>main</code>	669
18.161.2.2 <code>Test1</code>	669
18.162 <code>mtk_lapack_adapter_test.cc</code>	670
18.163 <code>tests/mtk_matrix_test/mtk_matrix_test.cc</code> File Reference	671
18.163.1Detailed Description	671
18.163.2Function Documentation	671
18.163.2.1 <code>main</code>	671
18.163.2.2 <code>TestDefaultConstructorGetters</code>	672
18.163.2.3 <code>TestIncreaseDecreaseNumNull</code>	674
18.163.2.4 <code>TestIncreaseDecreaseNumZero</code>	676
18.163.2.5 <code>TestSettersGettersCopyConstructor</code>	678
18.164 <code>mtk_matrix_test.cc</code>	680
18.165 <code>tests/mtk_robin_bc_descriptor_2d_test/mtk_robin_bc_descriptor_2d_test.cc</code> File Reference	684
18.165.1Detailed Description	684
18.165.2Function Documentation	685
18.165.2.1 <code>cc</code>	685
18.165.2.2 <code>HomogeneousDiricheletBC</code>	685
18.165.2.3 <code>main</code>	686
18.165.2.4 <code>ScalarField</code>	687

18.165.2.5TestDefaultConstructorGetters	688
18.165.2.6TestImposeOnGrid	689
18.165.2.7TestPushBackImposeOnLaplacianMatrix	691
18.166mtk_robin_bc_descriptor_2d_test.cc	693
18.167Tests/mtk_uni_stg_grid_1d_test/mtk_uni_stg_grid_1d_test.cc File Reference	695
18.167.1Detailed Description	695
18.167.2Function Documentation	696
18.167.2.1main	696
18.167.2.2ScalarField	696
18.167.2.3TestBindScalarFieldWriteToFile	697
18.167.2.4TestBindVectorField	698
18.167.2.5TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField	699
18.167.2.6TestDefaultConstructor	700
18.167.2.7VectorFieldPComponent	700
18.168mtk_uni_stg_grid_1d_test.cc	701
18.169Tests/mtk_uni_stg_grid_2d_test/mtk_uni_stg_grid_2d_test.cc File Reference	703
18.169.1Detailed Description	703
18.169.2Function Documentation	703
18.169.2.1main	703
18.169.2.2ScalarField	704
18.169.2.3TestBindScalarFieldWriteToFile	705
18.169.2.4TestBindVectorField	706
18.169.2.5TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSStreamOperator	706
18.169.2.6TestDefaultConstructor	707
18.169.2.7TestGetters	708
18.169.2.8VectorFieldPComponent	709
18.169.2.9VectorFieldQComponent	710
18.170mtk_uni_stg_grid_2d_test.cc	710
18.171Tests/mtk_uni_stg_grid_3d_test/mtk_uni_stg_grid_3d_test.cc File Reference	713
18.171.1Detailed Description	713
18.171.2Function Documentation	713
18.171.2.1main	713
18.171.2.2ScalarField	714
18.171.2.3TestBindScalarFieldWriteToFile	715
18.171.2.4TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSStreamOperator	716
18.171.2.5TestDefaultConstructor	716
18.171.2.6TestGetters	717

18.17 <code>mtk_uni_stg_grid_3d_test.cc</code>	719
Index	722

Chapter 1

Introduction

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

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

1.1 MTK Concerns

Since collaborative development efforts are very important, we have divided the library's source code according to the purpose the classes possess. These divisions (or **concerns**) are grouped by layers, and are hierarchically related by the dependencies 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. Foundations.
2. Enumerations.
3. Execution Tools.
4. Data Structures.
5. Numerical Methods.
6. Grids.
7. Mimetic Operators.

1.2 MTK Wrappers

The MTK collection of wrappers is:

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

Others are being considered.

1.3 Contact, Support and Credits

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

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

Currently the developers are:

- **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - ejspeiro
- Jose E. Castillo, PhD - jcastillo at mail dot sdsu dot edu
- Guillermo F. Miranda, PhD - unigrav at hotmail dot com

1.3.1 Acknowledgements and Contributions

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

1. Mohammad Abouali, Ph.D.
2. Dany De Cecchis, Ph.D.
3. Otilio Rojas, Ph.D.
4. Julia Rossi, Ph.D.
5. Christopher P. Paolini, PhD - paolini at engineering dot sdsu dot edu
6. Johnny Corbino.
7. Raul Vargas-Navarro.

Chapter 2

Referencing This Work

Please reference this work as follows:

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

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


Chapter 3

Read Me File and Installation Instructions

```
# The Mimetic Methods Toolkit (MTK)
```

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

1. Description

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

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

2. Dependencies

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

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

For OSX:

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

3. Installation

PART 1. CONFIGURATION OF THE MAKEFILE.

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

```
```
$ make help

Makefile for the MTK.

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

- testall: runs all the tests.

- gendoc: generates the documentation for the library.

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

```
```

```

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

```
```
$ make
```

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

```

4. Contact, Support, and Credits

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

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

Currently the developers are:

- **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - @ejspeiro
- Jose E. Castillo, PhD - jcastillo at mail dot sdsu dot edu
- Guillermo F. Miranda, PhD - unigrav at hotmail dot com

4.1. Acknowledgements and Contributions

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

- Mohammad Abouali, PhD
- Dany De Cecchis, PhD
- Otilio Rojas, PhD
- Julia Rossi, PhD
- Christopher P. Paolini, PhD - paolini at engineering dot sdsu dot edu
- Johnny Corbino.

- Raul Vargas-Navarro.

5. Referencing This Work

Please reference this work as follows:

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

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

```

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

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

Thanks and happy coding!

Chapter 4

Programming Tools

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

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

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

Chapter 5

Tests and Test Architectures

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

The MTK is intended to be as portable as possible throughout different architectures. The following architectures have provided flawless installations of the library and correct execution of the tests and the examples:

1. Intel(R) Pentium(R) M CPU 1.73 GHz 2048 KB of cache and stepping of 8.
Linux 3.2.0-23-generic-pae #36-Ubuntu SMP i386 GNU/Linux
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
2. Intel(R) Core(TM) i7-4700MQ CPU 2.40 GHz 6144 KB of cache and stepping of 3.
Linux 3.13.0-67-generic #110-Ubuntu SMP x86_64 GNU/Linux
gcc version 4.8.4 (Ubuntu 4.4.4-2ubuntu1~14.04)
3. Intel(R) Core(TM) i7-4600U CPU 2.10 GHz 4096 KB of cache and a stepping of 1.
Linux 3.16.7-29-desktop #1 SMP PREEMPT (6be6a97) x86_64 GNU/Linux
openSUSE 13.2 (Harlequin) (x86_64)
gcc (SUSE Linux) 4.8.3 20140627 [gcc-4_8-branch revision 212064]

Further architectures will be tested.

Chapter 6

User Manual, References and Theory

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

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

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

Chapter 7

Examples

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

Chapter 8

Definitions and Declarations

Throughout the library, we emphasize the difference between definitions and declarations as explained in: <http://www.lurklurk.org/linkers/linkers.html#cfile>

A definition associates a name with an implementation of that name, which could be either data or code (...)

A declaration tells the C compiler that a definition of something (with a particular name) exists elsewhere in the program, probably in a different C file. (Note that a definition also counts as a declaration—it's a declaration that also happens to fill in the particular "elsewhere").

Chapter 9

Licensing and Modifications

Copyright (C) 2016, Computational Science Research Center, San Diego State University. All rights reserved.

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

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

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

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

Chapter 10

Todo List

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

Implement Kronecker product using the BLAS.

Implement Kron using the BLAS.

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

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

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

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

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

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

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

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

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

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

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

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

Member `mtk::Tools::BeginUnitTestNo (const int &nn) noexcept`

Compute time using C++11 mechanisms.

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

Check if this is the best way of stalling execution in C++11.

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

Review const-correctness of the pointer we return.

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

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

Member `mtk::UniStgGrid2D::discrete_domain_x () const`

Review const-correctness of the pointer we return.

Member `mtk::UniStgGrid2D::discrete_domain_y () const`

Review const-correctness of the pointer we return.

Member [mtk::UniStgGrid3D::discrete_domain_x \(\) const](#)

Review const-correctness of the pointer we return.

Member [mtk::UniStgGrid3D::discrete_domain_y \(\) const](#)

Review const-correctness of the pointer we return.

Member [mtk::UniStgGrid3D::discrete_domain_z \(\) const](#)

Review const-correctness of the pointer we return.

File [mtk blas adapter.cc](#)

Write documentation using LaTeX.

File [mtk div 1d.cc](#)

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

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

File [mtk foundations.h](#)

Test selective precision mechanisms.

File [mtk glpk adapter test.cc](#)

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

File [mtk grad 1d.cc](#)

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

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

File [mtk lapack adapter.cc](#)

Write documentation using LaTeX.

File [mtk lapack adapter test.cc](#)

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

File [mtk quad 1d.h](#)

Implement this class.

File [mtk 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.

Create overloaded binding routines that read data from arrays.

File [mtk uni stg grid 3d.h](#)

Create overloaded binding routines that read data from files.

Create overloaded binding routines that read data from arrays.

Chapter 11

Module Index

11.1 Modules

Here is a list of all modules:

Foundations.	33
Enumerations.	36
Execution Tools.	38
Data Structures.	39
Numerical Methods.	40
Grids.	41
Mimetic Operators.	42

Chapter 12

Namespace Index

12.1 Namespace List

Here is a list of all namespaces with brief descriptions:

mtk	Mimetic Methods Toolkit namespace	45
---------------------	---	----

Chapter 13

Class Index

13.1 Class List

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

mtk::BLASAdapter	Adapter class for the BLAS API	55
mtk::Curl2D	Implements a 2D mimetic curl operator	65
mtk::DenseMatrix	Defines a common dense matrix, using a 1D array	69
mtk::Div1D	Implements a 1D mimetic divergence operator	91
mtk::Div2D	Implements a 2D mimetic divergence operator	106
mtk::Div3D	Implements a 3D mimetic divergence operator	110
mtk::GLPKAdapter	Adapter class for the GLPK API	115
mtk::Grad1D	Implements a 1D mimetic gradient operator	117
mtk::Grad2D	Implements a 2D mimetic gradient operator	133
mtk::Grad3D	Implements a 3D mimetic gradient operator	137
mtk::Interp1D	Implements a 1D interpolation operator	142
mtk::Interp2D	Implements a 2D interpolation operator	145
mtk::Lap1D	Implements a 1D mimetic Laplacian operator	148
mtk::Lap2D	Implements a 2D mimetic Laplacian operator	158
mtk::Lap3D	Implements a 3D mimetic Laplacian operator	162
mtk::LAPACKAdapter	Adapter class for the LAPACK API	167
mtk::Matrix	Definition of the representation of a matrix in the MTK	174

mtk::OperatorApplicator	Controls the process of applying a mimetic operator to a grid	205
mtk::Quad1D	Implements a 1D mimetic quadrature	208
mtk::RobinBCDescriptor1D	Impose Robin boundary conditions on the operators and on the grids	210
mtk::RobinBCDescriptor2D	Impose Robin boundary conditions on the operators and on the grids	220
mtk::RobinBCDescriptor3D	Impose Robin boundary conditions on the operators and on the grids	244
mtk::Tools	Tool manager class	253
mtk::UniStgGrid1D	Uniform 1D Staggered Grid	260
mtk::UniStgGrid2D	Uniform 2D Staggered Grid	273
mtk::UniStgGrid3D	Uniform 3D Staggered Grid	290

Chapter 14

File Index

14.1 File List

Here is a list of all files with brief descriptions:

examples/1d_accuracy/1d_accuracy.cc	Check the accuracy of mimetic operators	309
examples/1d_divergence/1d_divergence.cc	Creates instances of a 1D divergence as computed by the CBS algorithm	314
examples/1d_gradient/1d_gradient.cc	Creates instances of a 1D gradient as computed by the CBS algorithm	316
examples/1d_laplacian/1d_laplacian.cc	Creates instances of a 1D Laplacian as computed by the CBS algorithm	318
examples/1d_mimetic_measure/1d_mimetic_measure.cc	Compute the mimetic measure of different mimetic operators	320
examples/1d_mimetic_threshold/1d_mimetic_threshold.cc	Perform a sensitivity analysis of the mimetic threshold	322
examples/1d_poisson/1d_poisson.cc	Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	325
examples/1d_poisson_minimal/1d_poisson_minimal.cc	Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	333
examples/1d_poisson_sensitivity_threshold/1d_poisson_sensitivity_threshold.cc	Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	339
examples/1d_poisson_supercritical/1d_poisson_supercritical.cc	Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs	347
examples/1d_positive_weights/1d_positive_weights.cc	The CBS algorithm computes positive-definite weights, for 1D operators	354
examples/2d_angular_velocity/2d_angular_velocity.cc	Compute the curl of a 2D angular velocity field	357
examples/2d_poisson/2d_poisson.cc	Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs	359
examples/3d_diffusion/3d_diffusion.cc	Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs	363
include/mtk.h	Includes the entire API	366
include/mtk blas adapter.h	Declaration of an adapter class for the BLAS API	369
include/mtk_curl_2d.h	Includes the definition of the class Curl2D	371

include/mtk_dense_matrix.h	Declaration of a class dense matrix implemented using a 1D array	373
include/mtk_div_1d.h	Includes the definition of the class Div1D	376
include/mtk_div_2d.h	Includes the definition of the class Div2D	379
include/mtk_div_3d.h	Includes the definition of the class Div3D	381
include/mtk_enums.h	Definitions of the enumeration types in the MTK	383
include/mtk_foundations.h	Fundamental definitions to be used across all classes of the MTK	385
include/mtk_glpk_adapter.h	Declaration of an adapter class for the GLPK API	387
include/mtk_grad_1d.h	Includes the definition of the class Grad1D	390
include/mtk_grad_2d.h	Includes the definition of the class Grad2D	392
include/mtk_grad_3d.h	Includes the definition of the class Grad3D	394
include/mtk_interp_1d.h	Includes the definition of the class Interp1D	397
include/mtk_interp_2d.h	Includes the definition of the class Interp2D	399
include/mtk_lap_1d.h	Includes the definition of the class Lap1D	401
include/mtk_lap_2d.h	Includes the implementation of the class Lap2D	403
include/mtk_lap_3d.h	Includes the implementation of the class Lap3D	405
include/mtk_lapack_adapter.h	Declaration of an adapter class for the LAPACK API	407
include/mtk_matrix.h	Declaration of the representation of a matrix in the MTK	409
include/mtk_operator_applicator.h	Controls the process of applying a mimetic operator to a grid	413
include/mtk_quad_1d.h	Includes the definition of the class Quad1D	415
include/mtk_robin_bc_descriptor_1d.h	Impose Robin boundary conditions on the operators and on the grids	417
include/mtk_robin_bc_descriptor_2d.h	Impose Robin boundary conditions on the operators and on the grids	420
include/mtk_robin_bc_descriptor_3d.h	Impose Robin boundary conditions on the operators and on the grids	424
include/mtk_tools.h	Declaration of a class to manage run-time tools	427
include/mtk_uni_stg_grid_1d.h	Definition of an 1D uniform staggered grid	429
include/mtk_uni_stg_grid_2d.h	Definition of an 2D uniform staggered grid	432
include/mtk_uni_stg_grid_3d.h	Definition of an 3D uniform staggered grid	435
src/mtk_blas_adapter.cc	Definition of an adapter class for the BLAS API	441

src/mtk_curl_2d.cc	Implements the class Curl2D	447
src/mtk_dense_matrix.cc	449
src/mtk_div_1d.cc	Implements the class Div1D	458
src/mtk_div_2d.cc	Implements the class Div2D	477
src/mtk_div_3d.cc	Implements the class Div3D	480
src/mtk_glpk_adapter.cc	Definition of an adapter class for the GLPK API	483
src/mtk_grad_1d.cc	Implements the class Grad1D	488
src/mtk_grad_2d.cc	Implements the class Grad2D	509
src/mtk_grad_3d.cc	Implements the class Grad3D	512
src/mtk_interp_1d.cc	Includes the implementation of the class Interp1D	514
src/mtk_lap_1d.cc	Includes the implementation of the class Lap1D	518
src/mtk_lap_2d.cc	Includes the implementation of the class Lap2D	523
src/mtk_lap_3d.cc	Includes the implementation of the class Lap3D	525
src/mtk_lapack_adapter.cc	Definition of an adapter class for the LAPACK API	528
src/mtk_matrix.cc	Definition of the representation of a matrix in the MTK	537
src/mtk_operator_applicator.cc	Implements the class OperatorApplicator	544
src/mtk_robin_bc_descriptor_1d.cc	Impose Robin boundary conditions on the operators and on the grids	546
src/mtk_robin_bc_descriptor_2d.cc	Impose Robin boundary conditions on the operators and on the grids	550
src/mtk_tools.cc	Definition of a class to manage run-time tools	560
src/mtk_uni_stg_grid_1d.cc	Implementation of an 1D uniform staggered grid	562
src/mtk_uni_stg_grid_2d.cc	Implementation of a 2D uniform staggered grid	568
src/mtk_uni_stg_grid_3d.cc	Implementation of a 3D uniform staggered grid	575
tests/mtk blas_adapter_test/mtk blas_adapter_test.cc	Test file for the <code>mtk::BLASAdapter</code> class	582
tests/mtk_dense_matrix_test/mtk_dense_matrix_test.cc	Test file for the <code>mtk::DenseMatrix</code> class	585
tests/mtk_div_1d_test/mtk_div_1d_test.cc	Testing the mimetic 1D divergence, constructed with the CBS algorithm	597
tests/mtk_div_2d_test/mtk_div_2d_test.cc	Test file for the <code>mtk::Div2D</code> class	609
tests/mtk_div_3d_test/mtk_div_3d_test.cc	Test file for the <code>mtk::Div3D</code> class	614

tests/mtk_glpk_adapter_test/mtk_glpk_adapter_test.cc	619
Test file for the mtk::GLPKAdapter class	
tests/mtk_grad_1d_test/mtk_grad_1d_test.cc	621
Testing the mimetic 1D gradient, constructed with the CBS algorithm	
tests/mtk_grad_2d_test/mtk_grad_2d_test.cc	633
Test file for the mtk::Grad2D class	
tests/mtk_grad_3d_test/mtk_grad_3d_test.cc	638
Test file for the mtk::Grad3D class	
tests/mtk_interp_1d_test/mtk_interp_1d_test.cc	643
Testing the 1D interpolation	
tests/mtk_lap_1d_test/mtk_lap_1d_test.cc	647
Testing the 1D Laplacian operator	
tests/mtk_lap_2d_test/mtk_lap_2d_test.cc	658
Test file for the mtk::Lap2D class	
tests/mtk_lap_3d_test/mtk_lap_3d_test.cc	663
Test file for the mtk::Lap3D class	
tests/mtk_lapack_adapter_test/mtk_lapack_adapter_test.cc	668
Test file for the mtk::LAPACKAdapter class	
tests/mtk_matrix_test/mtk_matrix_test.cc	671
Unit test file for the mtk::Matrix class	
tests/mtk_robin_bc_descriptor_2d_test/mtk_robin_bc_descriptor_2d_test.cc	684
Test file for the mtk::RobinBCDescriptor2D class	
tests/mtk_uni_stg_grid_1d_test/mtk_uni_stg_grid_1d_test.cc	695
Test file for the mtk::UniStgGrid1D class	
tests/mtk_uni_stg_grid_2d_test/mtk_uni_stg_grid_2d_test.cc	703
Test file for the mtk::UniStgGrid2D class	
tests/mtk_uni_stg_grid_3d_test/mtk_uni_stg_grid_3d_test.cc	713
Test file for the mtk::UniStgGrid3D class	

Chapter 15

Module Documentation

15.1 Foundations.

Fundamental execution parameters and defined types.

Typedefs

- `typedef float mtk::Real`
Users can simply change this to build a double- or single-precision MTK.

Variables

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

15.1.1 Detailed Description

Fundamental execution parameters and defined types.

15.1.2 Typedef Documentation

15.1.2.1 mtk::Real

Warning

Defined as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

Definition at line 93 of file [mtk_foundations.h](#).

15.1.3 Variable Documentation

15.1.3.1 mtk::kCriticalOrderAccuracyDiv {8}

Definition at line 186 of file [mtk_foundations.h](#).

15.1.3.2 mtk::kCriticalOrderAccuracyGrad {10}

Definition at line 177 of file [mtk_foundations.h](#).

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

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

Definition at line 158 of file [mtk_foundations.h](#).

15.1.3.4 mtk::kDefaultOrderAccuracy {2}

Definition at line 168 of file [mtk_foundations.h](#).

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

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

Definition at line 143 of file [mtk_foundations.h](#).

15.1.3.6 mtk::kOne {1.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

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

15.1.3.7 mtk::kTwo {2.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

Definition at line 128 of file [mtk_foundations.h](#).

15.1.3.8 mtk::kZero {0.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined on Makefile.inc.

Definition at line 126 of file [mtk_foundations.h](#).

15.2 Enumerations.

Fundamental execution parameters and defined types.

Enumerations

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

15.2.1 Detailed Description

Fundamental execution parameters and defined types.

15.2.2 Enumeration Type Documentation

15.2.2.1 enum `mtk::DirInterp` [strong]

Used to tag different directions of interpolation supported.

Enumerator

`SCALAR_TO_VECTOR` Interpolations places scalar on vectors' location.

`VECTOR_TO_SCALAR` Interpolations places vectors on scalars' location.

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

15.2.2.2 enum `mtk::EncodedOperator` [strong]

Used to tag different different operators a matrix can encode.

Enumerator

`NOOP` No operator.

`GRADIENT` Gradient operator.

`DIVERGENCE` Divergence operator.

`INTERPOLATION` Interpolation operator.

CURL Curl operator.

LAPLACIAN Laplacian operator.

Definition at line 141 of file [mtkEnums.h](#).

15.2.2.3 enum mtk::FieldNature [strong]

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

See also

https://en.wikipedia.org/wiki/Scalar_field

https://en.wikipedia.org/wiki/Vector_field

Enumerator

SCALAR Scalar-valued field.

VECTOR Vector-valued field.

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

15.2.2.4 enum mtk::MatrixOrdering [strong]

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

See also

https://en.wikipedia.org/wiki/Row-major_order

Enumerator

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

COL_MAJOR Column-major ordering (Fortran).

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

15.2.2.5 enum mtk::MatrixStorage [strong]

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

Enumerator

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

BANDED Banded matrices ala LAPACK and ScalAPACK: Must be implemented.

CRS Compressed-Rows Storage: Must be implemented.

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

15.3 Execution Tools.

Tools to ensure execution correctness.

Classes

- class [mtk::Tools](#)

Tool manager class.

15.3.1 Detailed Description

Tools to ensure execution correctness.

15.4 Data Structures.

Fundamental data structures.

Classes

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

15.4.1 Detailed Description

Fundamental data structures.

15.5 Numerical Methods.

Adapter classes and auxiliary numerical methods.

Classes

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

15.5.1 Detailed Description

Adapter classes and auxiliary numerical methods.

15.6 Grids.

Uniform rectangular staggered grids.

Classes

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

15.6.1 Detailed Description

Uniform rectangular staggered grids.

15.7 Mimetic Operators.

Mimetic operators.

Classes

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

Typedefs

- `typedef Real(* mtk::CoefficientFunction0D)(const Real &tt, const std::vector< Real > &pp)`
A function of a BC coefficient evaluated on a 0D domain and time.
- `typedef Real(* mtk::CoefficientFunction1D)(const Real &xx, const Real &tt)`
A function of a BC coefficient evaluated on a 1D domain and time.
- `typedef Real(* mtk::CoefficientFunction2D)(const Real &xx, const Real &yy, const Real &tt)`
A function of a BC coefficient evaluated on a 2D domain and time.

15.7.1 Detailed Description

Mimetic operators.

15.7.2 Typedef Documentation

15.7.2.1 mtk::CoefficientFunction0D

Definition at line 108 of file [mtk_robin_bc_descriptor_1d.h](#).

15.7.2.2 mtk::CoefficientFunction1D

Definition at line 97 of file [mtk_robin_bc_descriptor_2d.h](#).

15.7.2.3 mtk::CoefficientFunction2D

Definition at line 97 of file [mtk_robin_bc_descriptor_3d.h](#).

Chapter 16

Namespace Documentation

16.1 mtk Namespace Reference

Mimetic Methods Toolkit namespace.

Classes

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

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

Typedefs

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

Enumerations

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

- enum `DirInterp` { `DirInterp::SCALAR_TO_VECTOR`, `DirInterp::VECTOR_TO_SCALAR` }
Interpolation operator.
- enum `EncodedOperator` {
`EncodedOperator::NOOP`, `EncodedOperator::GRADIENT`, `EncodedOperator::DIVERGENCE`, `EncodedOperator::INTERPOLATION`,
`EncodedOperator::CURL`, `EncodedOperator::LAPLACIAN` }
Operators matrices can encode.

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, int *ldb, double *beta, double *c, int *ldc)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::DenseMatrix` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::Div1D` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::Grad1D` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::Interp1D` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::Lap1D` &in)
- void `sgesv_` (int *n, int *nrhs, `Real` *a, int *lda, int *ipiv, `Real` *b, int *ldb, int *info)
- void `sgels_` (char *trans, int *m, int *n, int *nrhs, `Real` *a, int *lda, `Real` *b, int *ldb, `Real` *work, int *lwork, int *info)
Single-precision GEneral matrix Least Squares solver.
- void `sgeqrf_` (int *m, int *n, `Real` *a, int *lda, `Real` *tau, `Real` *work, int *lwork, int *info)
Single-precision GEneral matrix QR Factorization.
- void `sormqr_` (char *side, char *trans, int *m, int *n, int *k, `Real` *a, int *lda, `Real` *tau, `Real` *c, int *ldc, `Real` *work, int *lwork, int *info)
Single-precision Orthogonal Matrix from QR factorization.
- std::ostream & `operator<<` (std::ostream &stream, `mtk::UniStgGrid1D` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::UniStgGrid2D` &in)
- std::ostream & `operator<<` (std::ostream &stream, `mtk::UniStgGrid3D` &in)

Variables

- const float `kZero` {0.0f}
MTK's zero defined according to selective compilation.
- const float `kOne` {1.0f}
MTK's one defined according to selective compilation.
- const float `kTwo` {2.0f}
MTK's two defined according to selective compilation.
- const float `kDefaultTolerance` {1e-7f}
Considered tolerance for comparisons in numerical methods.
- const float `kDefaultMimeticThreshold` {1e-6f}
Default threshold for higher-order mimetic operators.
- const int `kDefaultOrderAccuracy` {2}
Default order of accuracy for mimetic operators.

- const int **kCriticalOrderAccuracyGrad** {10}
At this order (and higher) we must use the CBSA to construct gradients.
- const int **kCriticalOrderAccuracyDiv** {8}
At this order (and higher) we must use the CBSA to construct divergences.

16.1.1 Function Documentation

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

1. Print approximating coefficients for the interior.

Definition at line 66 of file [mtk_interp_1d.cc](#).

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

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

Definition at line 67 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



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

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

Definition at line 67 of file [mtk_uni_stg_grid_2d.cc](#).

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

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

Definition at line 68 of file [mtk_uni_stg_grid_1d.cc](#).

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

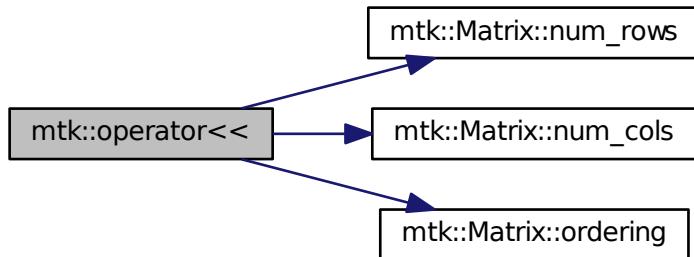
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

Definition at line 75 of file [mtk_lap_1d.cc](#).

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

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

Here is the call graph for this function:



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

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

Definition at line 82 of file [mtk_grad_1d.cc](#).

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

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

Definition at line 82 of file [mtk_div_1d.cc](#).

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

Here is the caller graph for this function:



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

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

The following options are provided:

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

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 >= 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

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

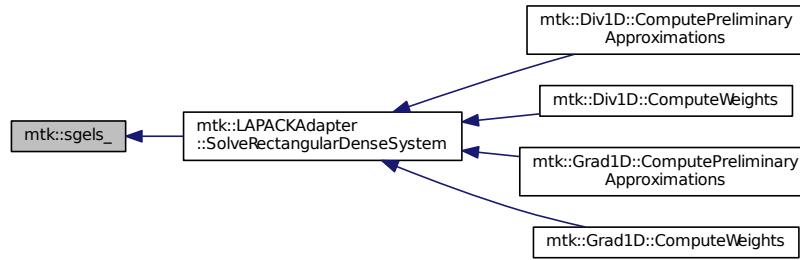
<http://www.math.utah.edu/software/lapack/lapack-s/sgels.html>

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 >= 0.
in	<i>n</i>	The number of columns of the matrix a. n >= 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 >= max(1,m).
in,out	<i>b</i>	On entry, matrix b of right-hand side vectors.

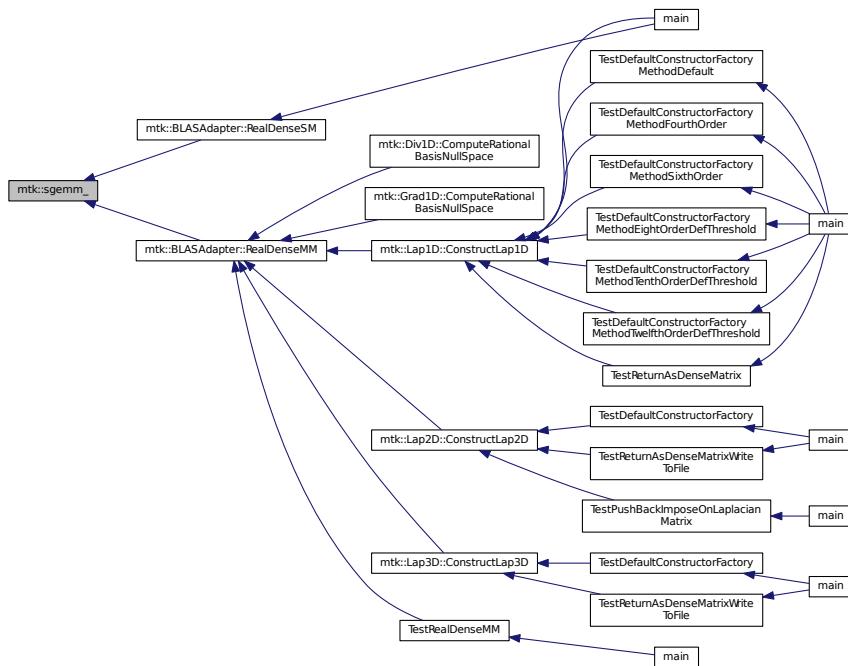
in	<i>ldb</i>	The leading dimension of b. $ldb \geq \max(1,m,n)$.
in,out	<i>work</i>	On exit, if info = 0, work(1) is optimal lwork.
in,out	<i>lwork</i>	The dimension of the array work.
in,out	<i>info</i>	If info = 0, then successful exit.

Here is the caller graph for this function:



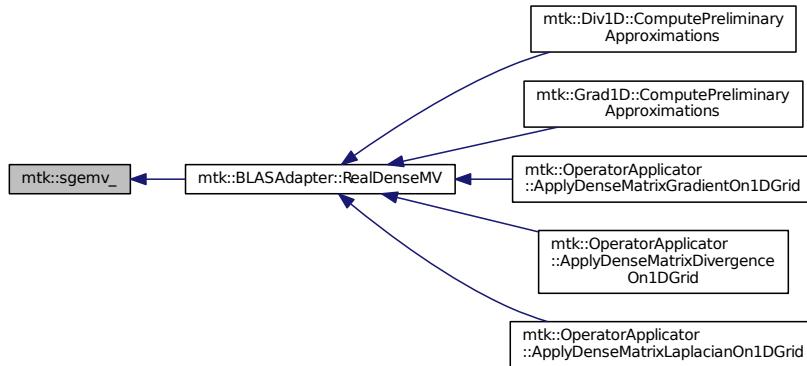
16.1.1.11 `void mtk::sgemm_(char * transa, char * transb, int * m, int * n, int * k, double * alpha, double * a, int * lda, double * b, int * ldb, double * beta, double * c, int * ldc)`

Here is the caller graph for this function:



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

Here is the caller graph for this function:



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

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

SIDE = 'L' SIDE = 'R'

TRANS = 'N': $Q \times C$ $C \times Q$ TRANS = 'T': $Q^T \times C$ $C \times Q^T$

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

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

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

See also

http://www.netlib.org/lapack/explore-html/df/d97/sgeqrf_8f.html

Parameters

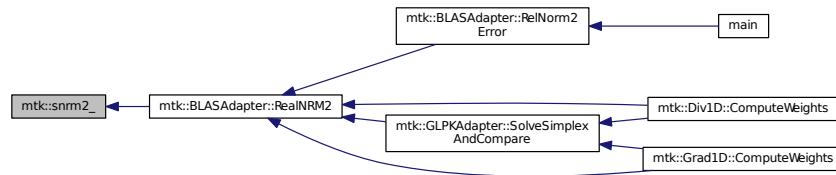
in	<i>m</i>	The number of columns of the matrix a. $n \geq 0$.
in	<i>n</i>	The number of columns of the matrix a. $n \geq 0$.
in,out	<i>a</i>	On entry, the n-by-n matrix a.
in	<i>lda</i>	Leading dimension matrix. $LDA \geq \max(1,M)$.
in,out	<i>tau</i>	Scalars from elementary reflectors. $\min(M,N)$.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

in	<i>lwork</i>	The dimension of work. <i>lwork</i> $\geq \max(1,n)$.
in	<i>info</i>	info = 0: successful exit.

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

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

Here is the caller graph for this function:



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

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

SIDE = 'L' SIDE = 'R'

TRANS = 'N': Q * C C * Q TRANS = 'T': Q**T * C C * Q**T

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

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

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

See also

http://www.netlib.org/lapack/explore-html/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.
in,out	<i>info</i>	info = 0: successful exit.

Chapter 17

Class Documentation

17.1 mtk::BLASAdapter Class Reference

Adapter class for the BLAS API.

```
#include <mtk blas adapter.h>
```

Collaboration diagram for mtk::BLASAdapter:

mtk::BLASAdapter
+ RealNRM2() + RealAXPY() + RelNorm2Error() + RealDenseMV() + RealDenseMV() + RealDenseMM() + RealDenseSM()

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 void `RealDenseMV (Real &alpha, Real *aa, MatrixOrdering &ordering, int num_rows, int num_cols, int lda, Real *xx, Real &beta, Real *yy)`

Real-Arithmetic General (Dense matrices) Matrix-Vector Multiplier.

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

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

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

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

17.1.1 Detailed Description

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

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

See also

<http://www.netlib.org/blas/>

<https://software.intel.com/en-us/non-commercial-software-development>

Definition at line 100 of file `mtk blas_adapter.h`.

17.1.2 Member Function Documentation

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

Performs

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

Parameters

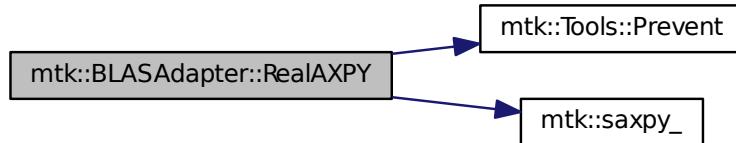
<code>in</code>	<code>alpha</code>	Scalar of the first array.
<code>in</code>	<code>xx</code>	First array.
<code>in</code>	<code>yy</code>	Second array.
<code>in</code>	<code>in_length</code>	Lengths of the given arrays.

Returns

Norm-2 of the given array.

Definition at line 342 of file [mtk blas adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Performs:

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

Parameters

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

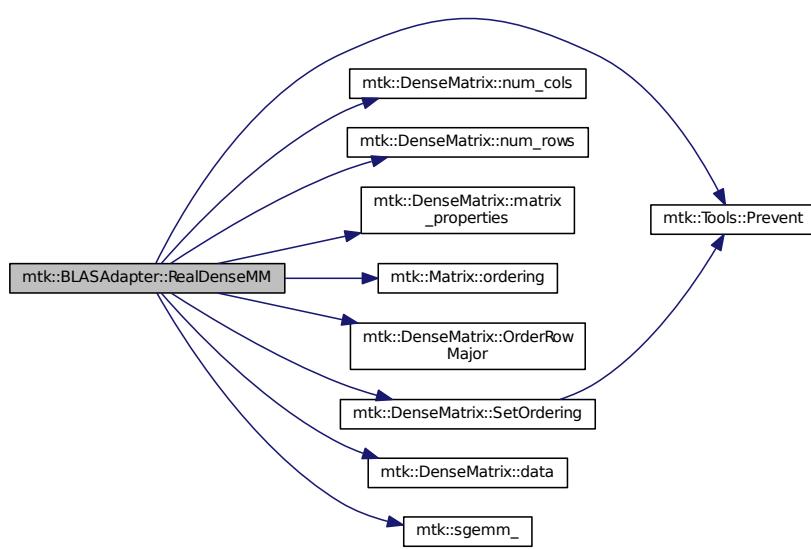
See also

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

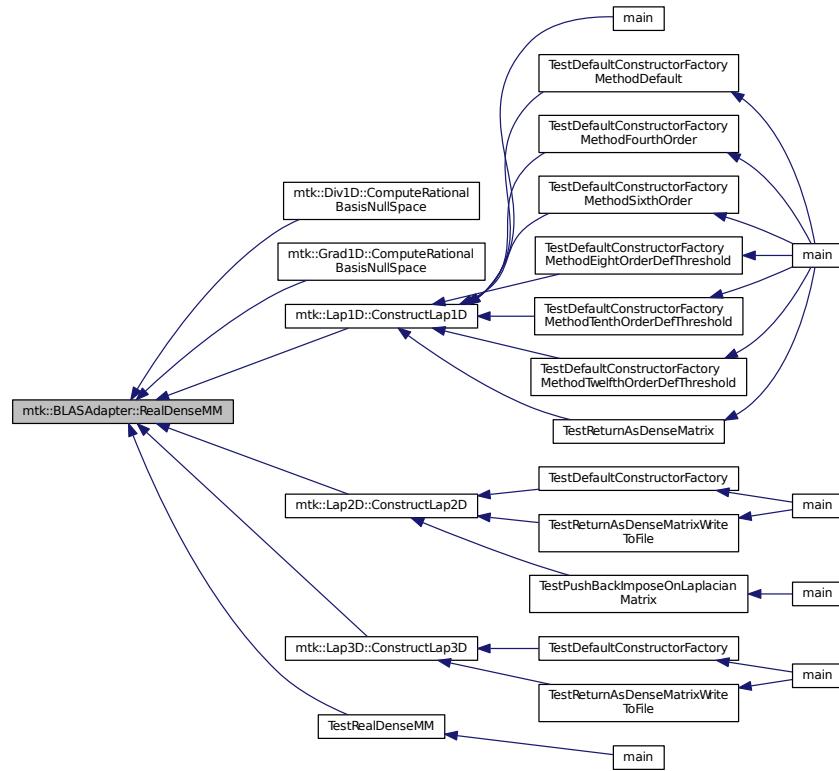
1. Make sure input matrices are row-major ordered.
2. Setup the problem.
3. Perform multiplication.

Definition at line 446 of file [mtk blas adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Performs

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

Parameters

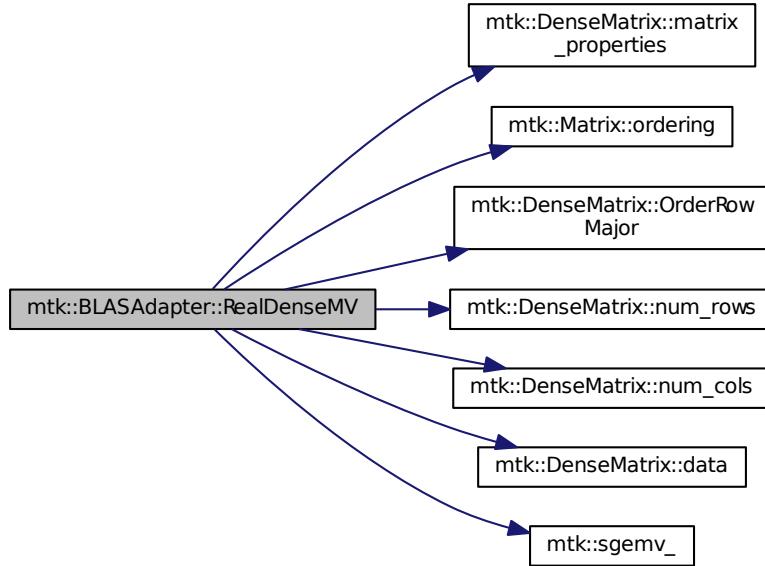
in	<i>alpha</i>	First scalar.
in	<i>aa</i>	Given matrix.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).

See also

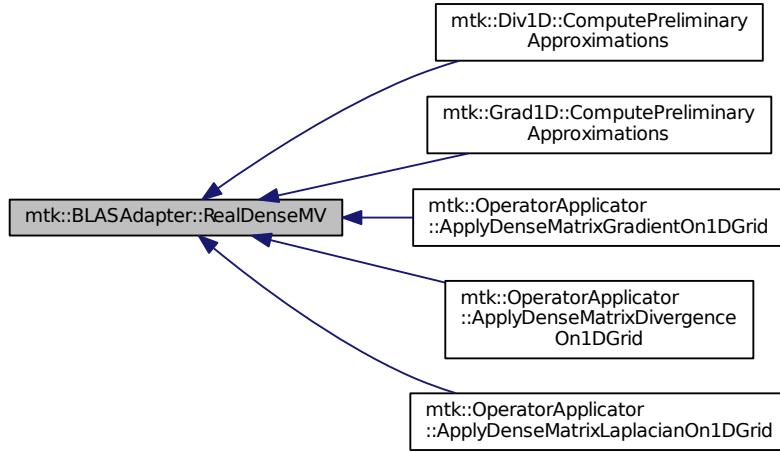
<http://ejspeiro.github.io/Netlib-and-CPP/>

Definition at line 381 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.1.2.4 void mtk::BLASAdapter::RealDenseMV (*mtk::Real & alpha*, *mtk::Real * aa*, *mtk::MatrixOrdering & ordering*, int *num_rows*, int *num_cols*, int *lda*, *mtk::Real * xx*, *mtk::Real & beta*, *mtk::Real * yy*) [static]

Performs

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

Parameters

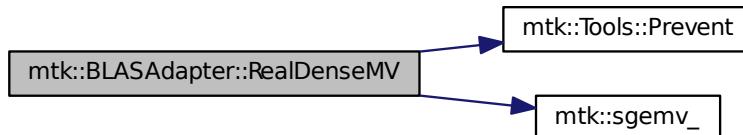
in	<i>alpha</i>	First scalar.
in	<i>aa</i>	Given matrix.
in	<i>ordering</i>	Ordering of the matrix.
in	<i>num_rows</i>	Number of rows.
in	<i>num_cols</i>	Number of columns.
in	<i>lda</i>	Leading dimension.
in	<i>xx</i>	First vector.
in	<i>beta</i>	Second scalar.
in, out	<i>yy</i>	Second vector (output).

See also

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

Definition at line 412 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



17.1.2.5 *mtk::DenseMatrix* mtk::BLASAdapter::RealDenseSM (*mtk::Real alpha*, *mtk::DenseMatrix & aa*) [static]

Performs:

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

Parameters

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

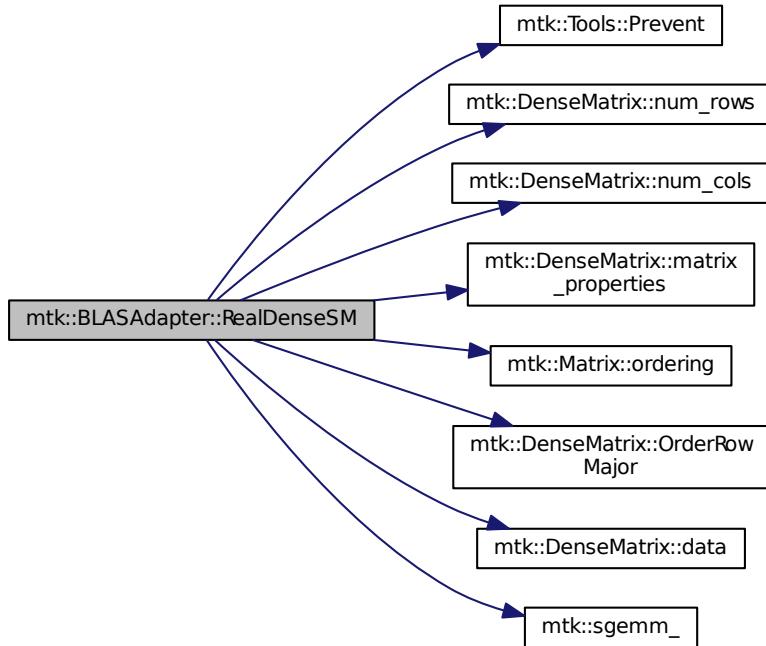
See also

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

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

Definition at line 503 of file [mtk blas adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

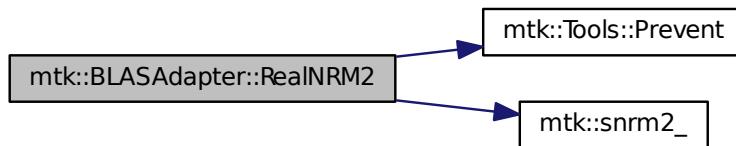
in	<i>in</i>	Input array.
in	<i>in_length</i>	Length of the array.

Returns

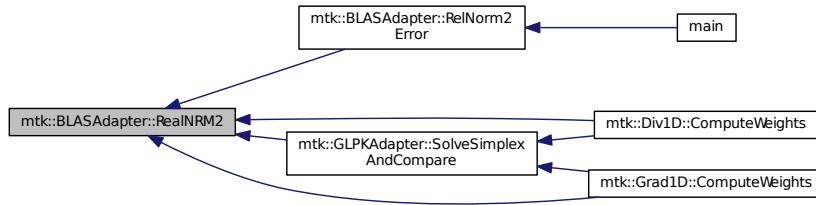
Norm-2 of the given array.

Definition at line 327 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.1.2.7 `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

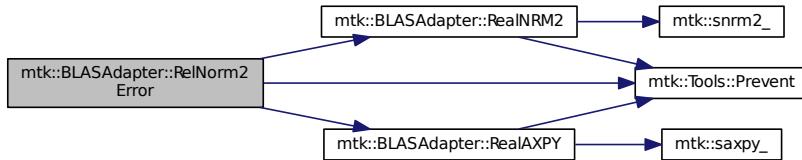
in	<i>known</i>	Array containing the computed solution.
in	<i>computed</i>	Array containing the known solution (ref. solution).

Returns

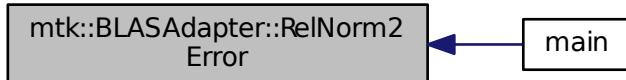
Relative norm-2 of the error, aka, the difference between the arrays.

Definition at line 361 of file [mtk blas 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:

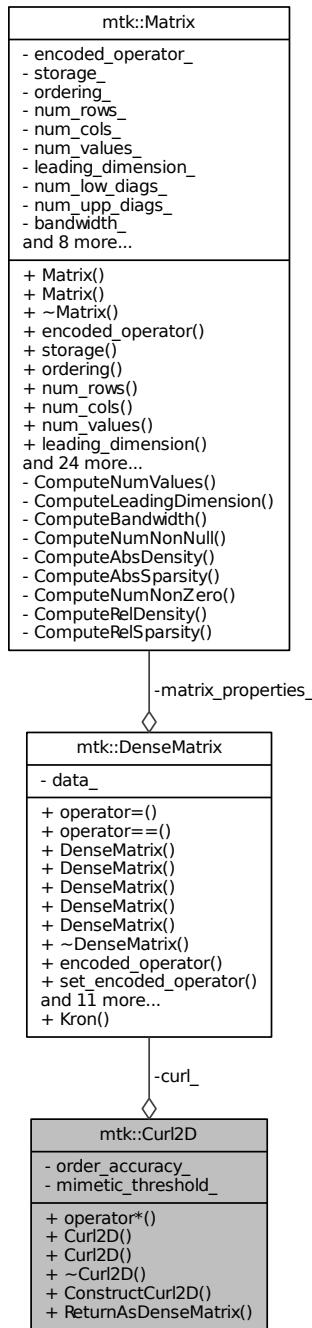
- [include/mtk blas adapter.h](#)
- [src/mtk blas adapter.cc](#)

17.2 mtk::Curl2D Class Reference

Implements a 2D mimetic curl operator.

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

Collaboration diagram for mtk::Curl2D:



Public Member Functions

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

- *Operator application operator on a grid.*
- `Curl2D ()`
Default constructor.
- `Curl2D (const Curl2D &curl)`
Copy constructor.
- `~Curl2D ()`
Destructor.
- `bool ConstructCurl2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`
Factory method implementing the CBS Algorithm to build operator.
- `DenseMatrix ReturnAsDenseMatrix () const`
Return the operator as a dense matrix.

Private Attributes

- `DenseMatrix curl_`
Actual operator.
- `int order_accuracy_`
Order of accuracy.
- `Real mimetic_threshold_`
Mimetic Threshold.

17.2.1 Detailed Description

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

Definition at line 77 of file [mtk_curl_2d.h](#).

17.2.2 Constructor & Destructor Documentation

17.2.2.1 mtk::Curl2D::Curl2D()

Definition at line 79 of file [mtk_curl_2d.cc](#).

17.2.2.2 mtk::Curl2D::Curl2D(const Curl2D & curl)

Parameters

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

Definition at line 83 of file [mtk_curl_2d.cc](#).

17.2.2.3 mtk::Curl2D::~Curl2D()

Definition at line 87 of file [mtk_curl_2d.cc](#).

17.2.3 Member Function Documentation

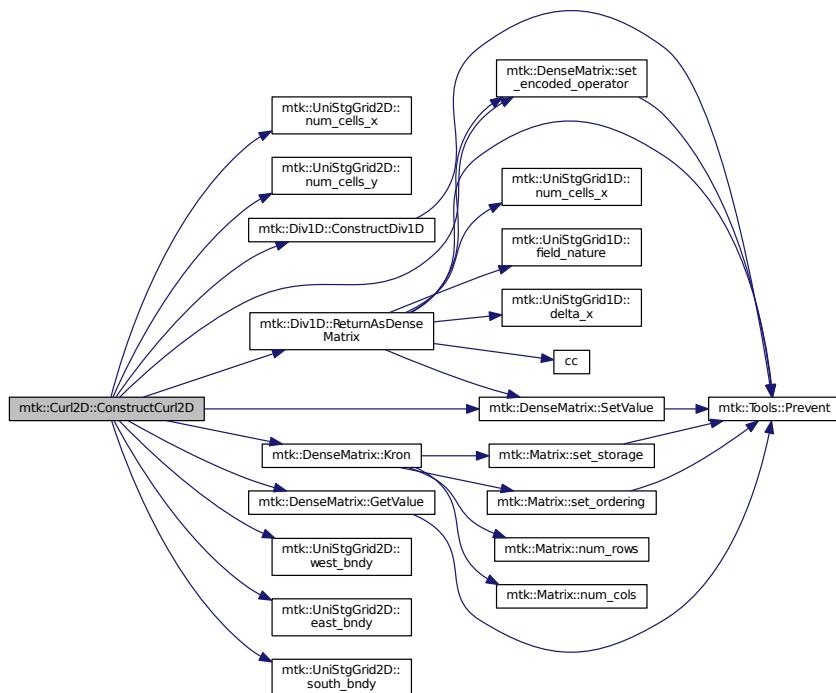
17.2.3.1 `bool mtk::Curl2D::ConstructCurl2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

Definition at line 89 of file [mtk_curl_2d.cc](#).

Here is the call graph for this function:



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

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

Definition at line 70 of file [mtk_curl_2d.cc](#).

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

Returns

The operator as a dense matrix.

Definition at line 159 of file [mtk_curl_2d.cc](#).

17.2.4 Member Data Documentation

17.2.4.1 DenseMatrix mtk::Curl2D::curl_ [private]

Definition at line 118 of file [mtk_curl_2d.h](#).

17.2.4.2 Real mtk::Curl2D::mimetic_threshold_ [private]

Definition at line 122 of file [mtk_curl_2d.h](#).

17.2.4.3 int mtk::Curl2D::order_accuracy_ [private]

Definition at line 120 of file [mtk_curl_2d.h](#).

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

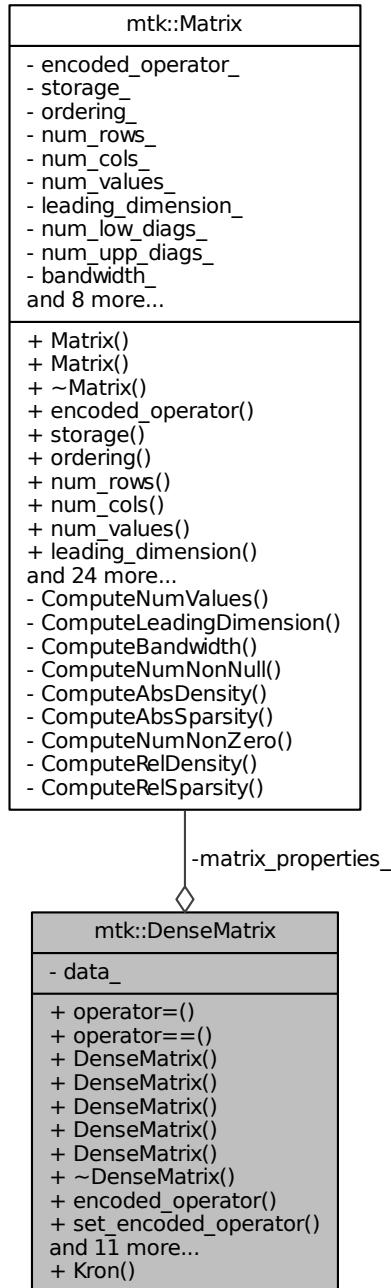
- [include/mtk_curl_2d.h](#)
- [src/mtk_curl_2d.cc](#)

17.3 mtk::DenseMatrix Class Reference

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

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

Collaboration diagram for mtk::DenseMatrix:



Public Member Functions

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

- Overloaded assignment operator.
- bool `operator==` (const `DenseMatrix` &in)
 - Overloaded comparison 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 *const gen, const int &gen_length, const int &pro_length, const bool &transpose)`
 - Construct a dense Vandermonde matrix.
- `~DenseMatrix ()`
 - Destructor.
- `EncodedOperator encoded_operator () const`
 - Return operator encoded by this matrix.
- void `set_encoded_operator (const EncodedOperator &op)`
 - Sets the encoded operator.
- `Matrix matrix_properties () const noexcept`
 - Provides access to the matrix data.
- int `num_rows () const noexcept`
 - Gets the number of rows.
- int `num_cols () const noexcept`
 - Gets the number of columns.
- `Real * data () const noexcept`
 - Provides access to the matrix value array.
- void `SetOrdering (mtk::MatrixOrdering oo) noexcept`
 - Sets the ordering of the matrix.
- `Real GetValue (const int &row_coord, const int &col_coord) const noexcept`
 - Gets a value on the given coordinates.
- void `SetValue (const int &row_coord, const int &col_coord, const Real &val) noexcept`
 - Sets a value on the given coordinates.
- void `Transpose ()`
 - Transpose this matrix.
- void `OrderRowMajor ()`
 - Make the matrix row-wise ordered.
- void `OrderColMajor ()`
 - Make the matrix column-wise ordered.
- bool `WriteToFile (const std::string &filename) const`
 - Writes matrix to a file compatible with Gnuplot 4.6.

Static Public Member Functions

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

Private Attributes

- `Matrix matrix_properties_`
Data related to the matrix nature.
- `Real * data_`
Array holding the data in contiguous position in memory.

Friends

- `std::ostream & operator<< (std::ostream &stream, DenseMatrix &in)`
Prints the matrix as a block of numbers (standard way).

17.3.1 Detailed Description

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

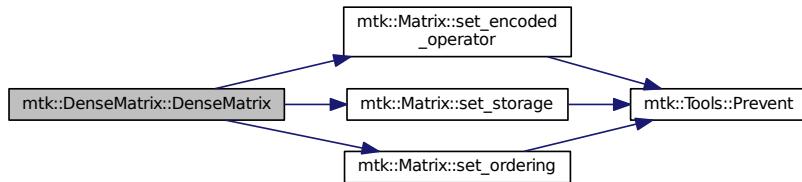
Definition at line 93 of file [mtk_dense_matrix.h](#).

17.3.2 Constructor & Destructor Documentation

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

Definition at line 173 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



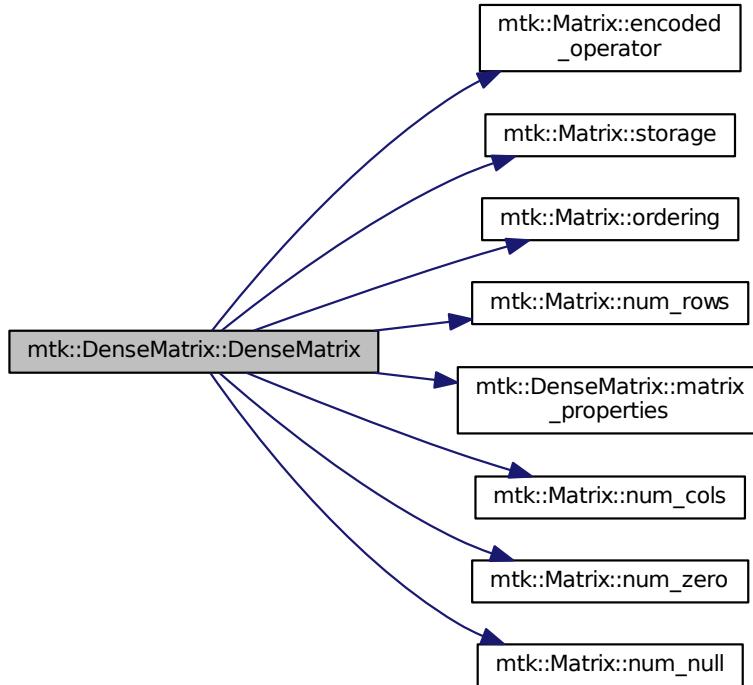
17.3.2.2 `mtk::DenseMatrix::DenseMatrix (const DenseMatrix & in)`

Parameters

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

Definition at line 181 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



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

Parameters

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

Exceptions

<code>std::bad_alloc</code>

Definition at line 217 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



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

Used in the construction of the mimetic operators.

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

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

Parameters

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

Exceptions

<i>std::bad_alloc</i>

Definition at line 241 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



17.3.2.5 mtk::DenseMatrix::DenseMatrix (const Real *const gen, const int & gen_length, const int & pro_length, const bool & transpose)

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

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

This constructor generates a Vandermonde matrix, as defined above.

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

Parameters

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

Exceptions

<i>std::bad_alloc</i>

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

Here is the call graph for this function:



17.3.2.6 mtk::DenseMatrix::~DenseMatrix()

Definition at line 334 of file [mtk_dense_matrix.cc](#).

17.3.3 Member Function Documentation

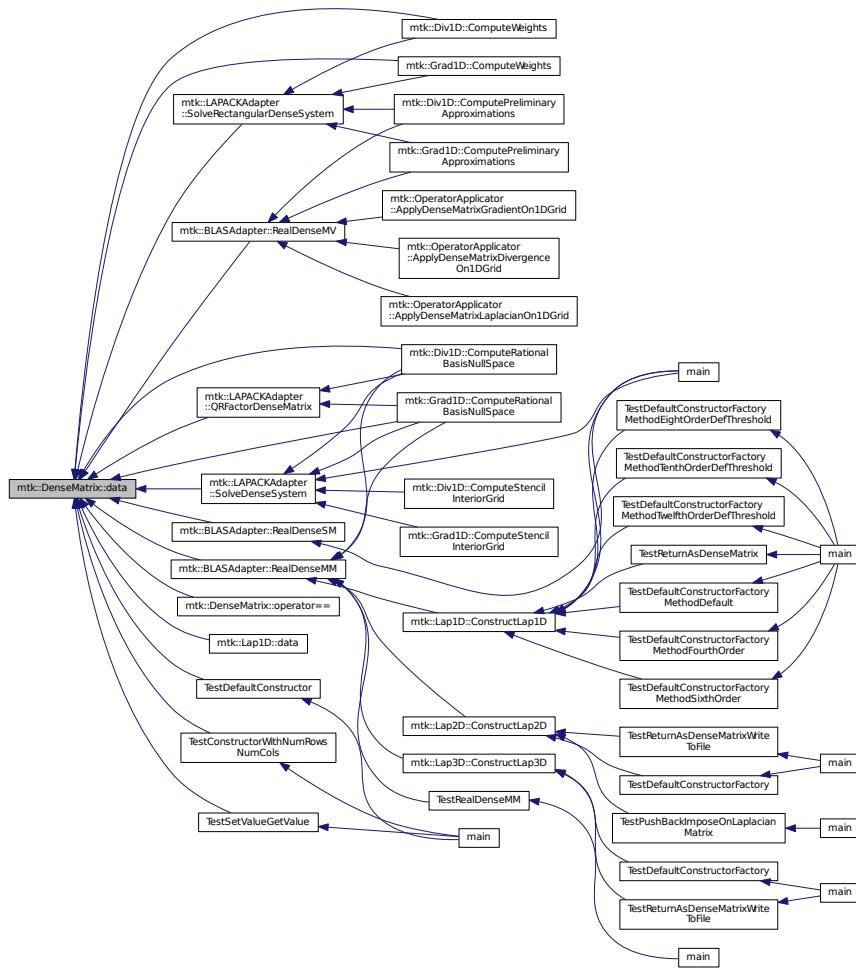
17.3.3.1 mtk::Real * mtk::DenseMatrix::data() const [noexcept]

Returns

Pointer to an array of `mtk::Real`.

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

Here is the caller graph for this function:



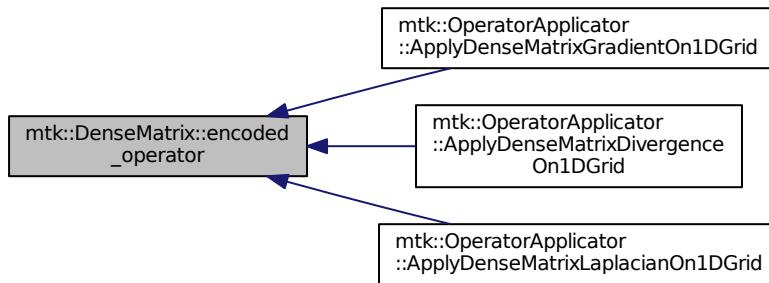
17.3.3.2 `mtk::EncodedOperator mtk::DenseMatrix::encoded_operator() const`

Returns

Operator encoded by this matrix.

Definition at line 371 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



17.3.3.3 mtk::Real mtk::DenseMatrix::GetValue (const int & *row_coord*, const int & *col_coord*) const [noexcept]

Parameters

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

Returns

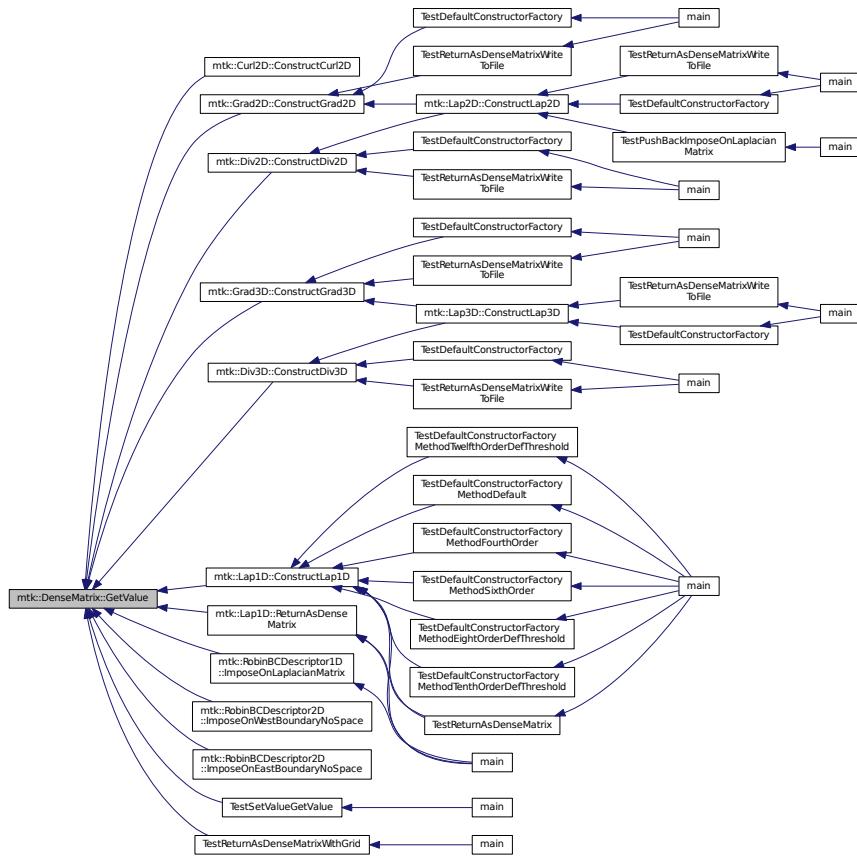
The required value at the specified coordinates.

Definition at line 392 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

in	aa	First matrix.
in	bb	Second matrix.

Exceptions

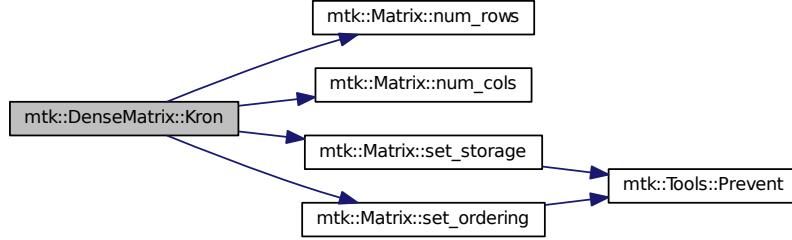
std::bad_alloc

Todo Implement Kronecker product using the BLAS.

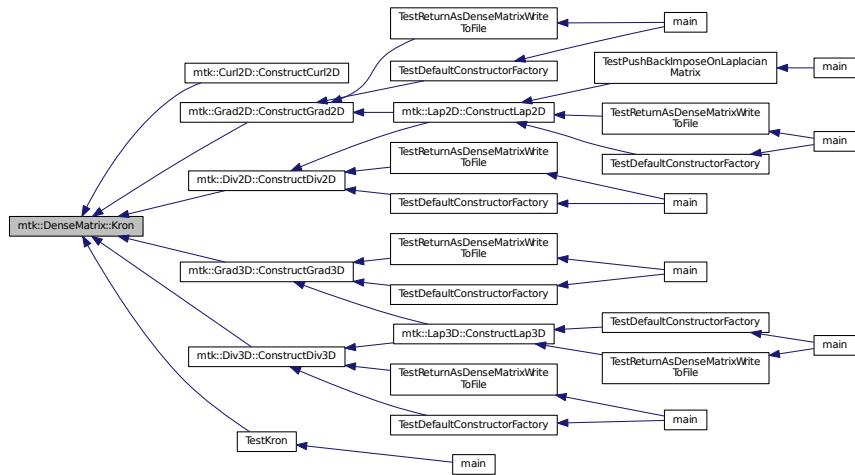
Todo Implement Kron using the BLAS.

Definition at line 534 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



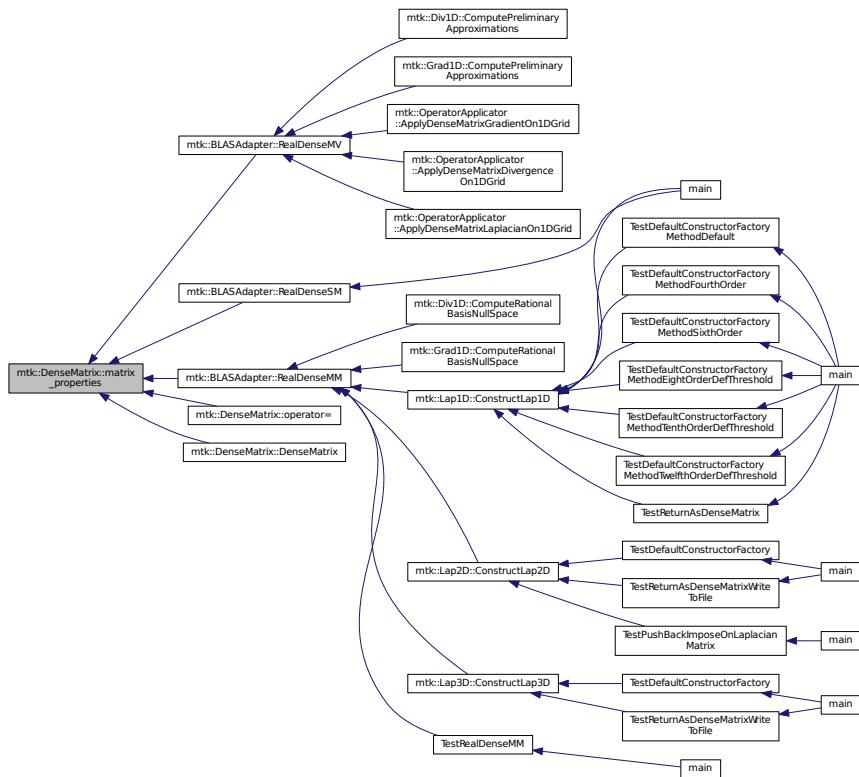
17.3.3.5 mtk::Matrix mtk::DenseMatrix::matrix_properties() const [noexcept]

Returns

Pointer to a [Matrix](#).

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

Here is the caller graph for this function:



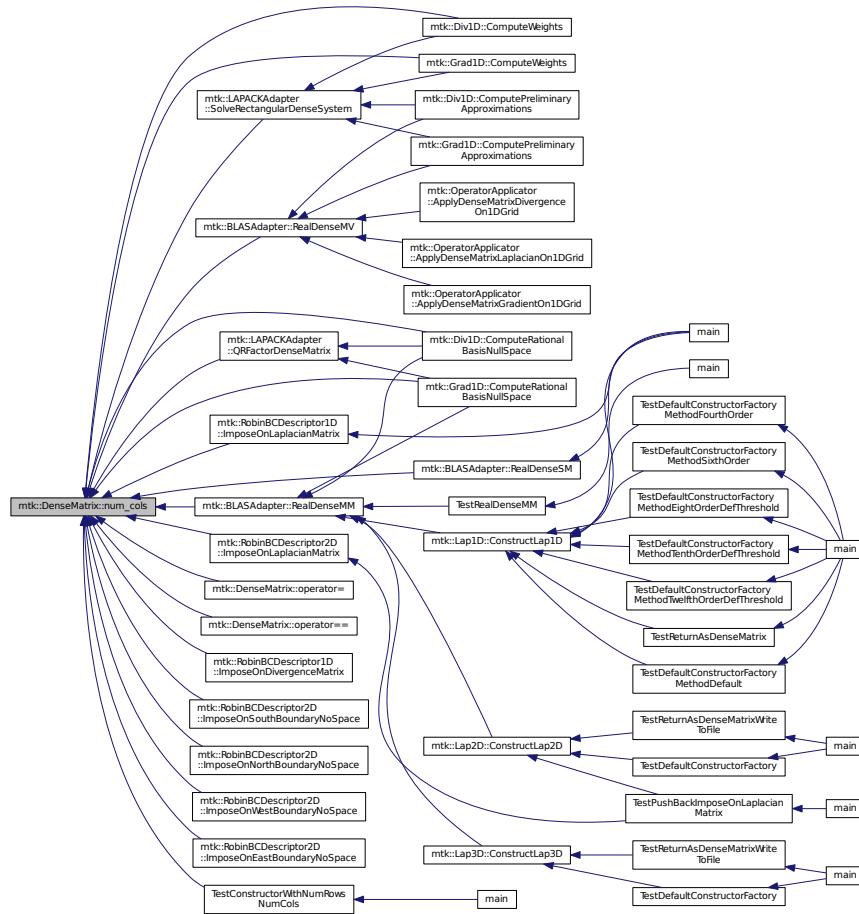
17.3.3.6 int mtk::DenseMatrix::num_cols() const [noexcept]

Returns

Number of columns of the matrix.

Definition at line 361 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



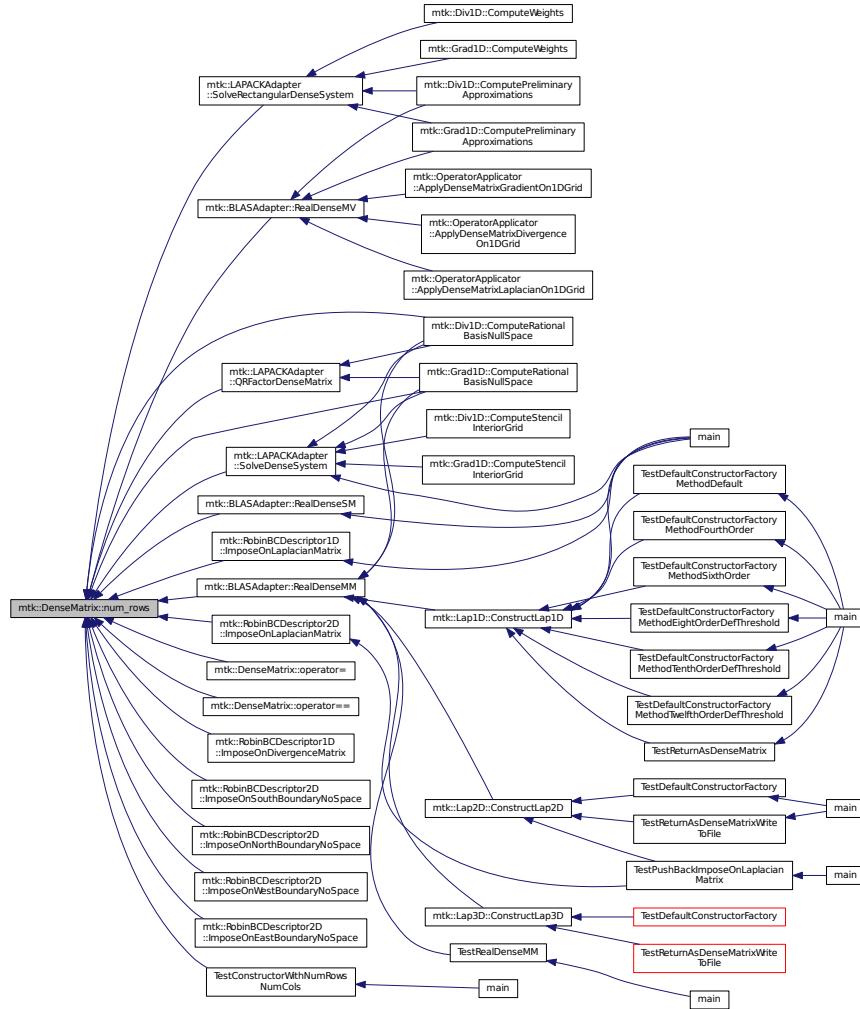
17.3.3.7 int mtk::DenseMatrix::num_rows () const [noexcept]

Returns

Number of rows of the matrix.

Definition at line 356 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



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

Parameters

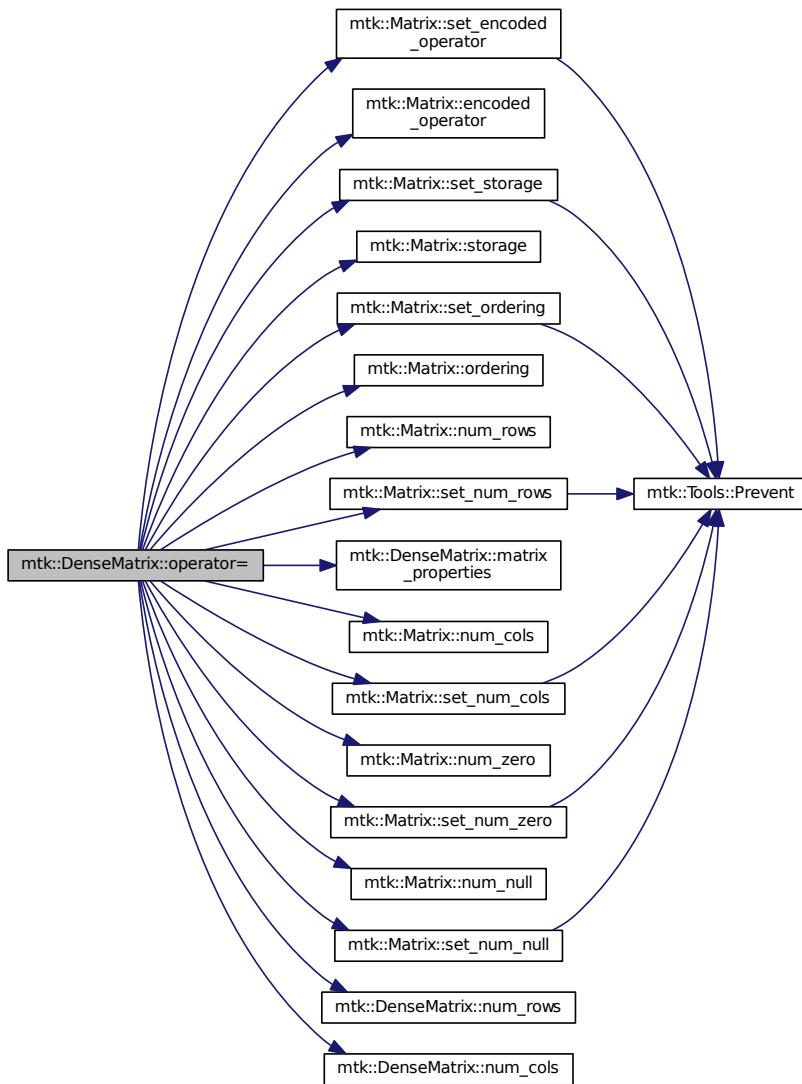
<code>in</code>	<code>in</code>	Given matrix.
-----------------	-----------------	---------------

Returns

Copy of the given matrix.

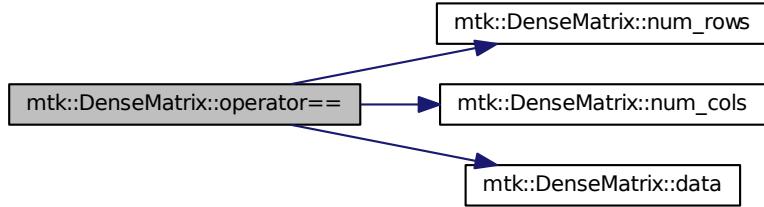
Definition at line 104 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:

**17.3.3.9 bool mtk::DenseMatrix::operator== (const DenseMatrix & in)**

Definition at line 148 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:

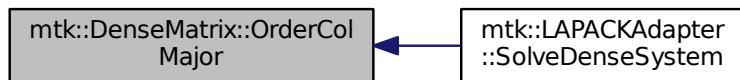


17.3.3.10 void mtk::DenseMatrix::OrderColMajor ()

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

Definition at line 495 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:

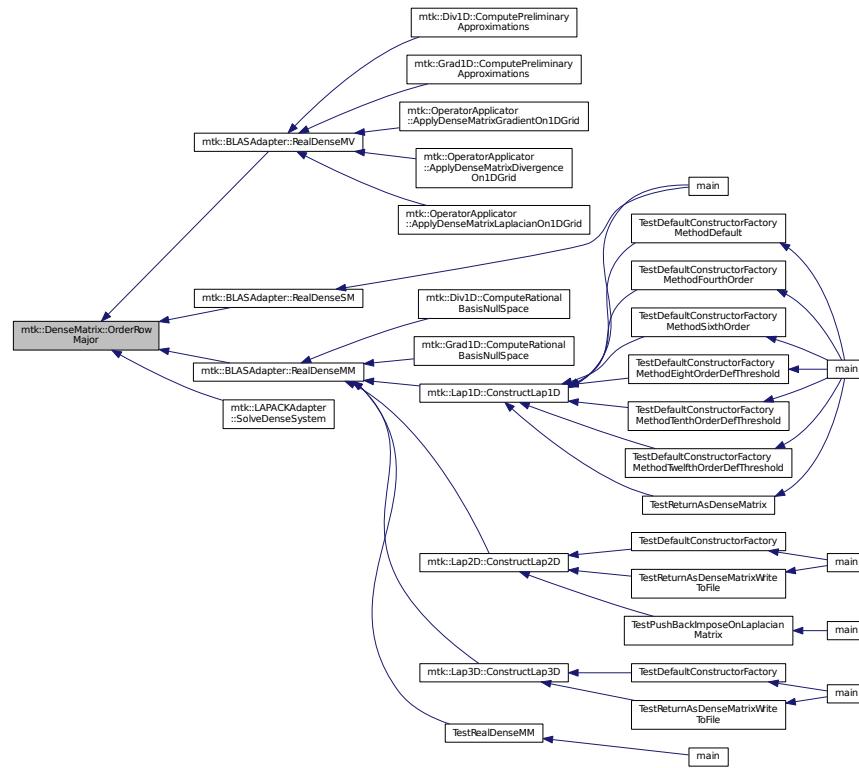


17.3.3.11 void mtk::DenseMatrix::OrderRowMajor ()

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

Definition at line 454 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



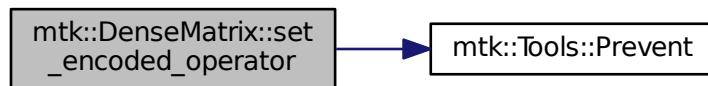
17.3.3.12 void mtk::DenseMatrix::set_encoded_operator (const EncodedOperator & op)

Parameters

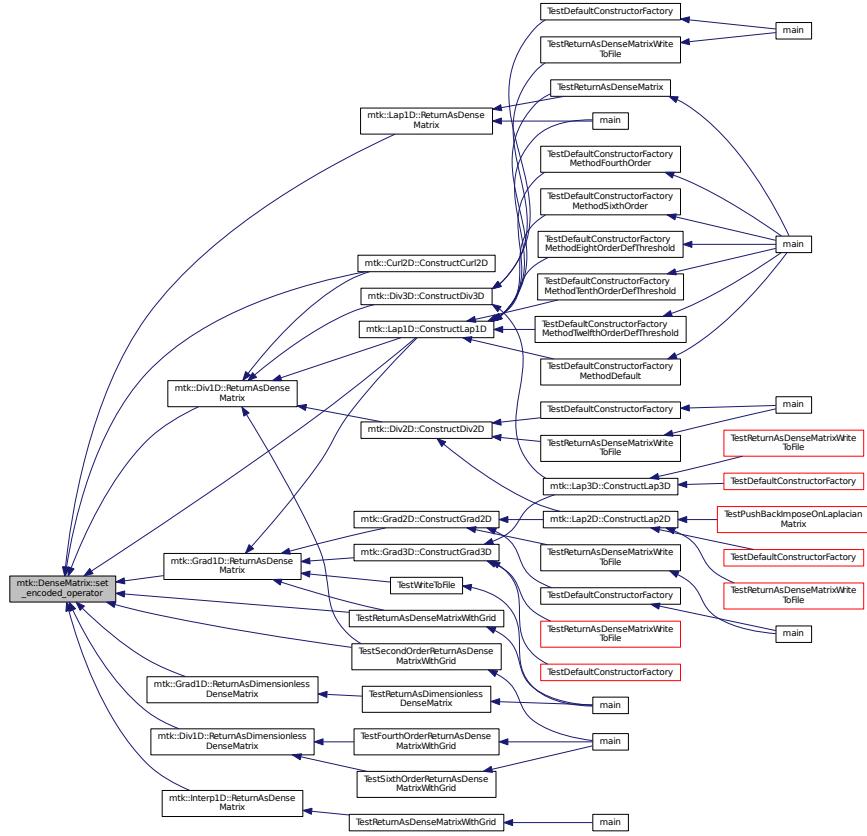
in	op	Encoded operator.
----	----	-------------------

Definition at line 376 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

in *oo* Ordering.

Returns

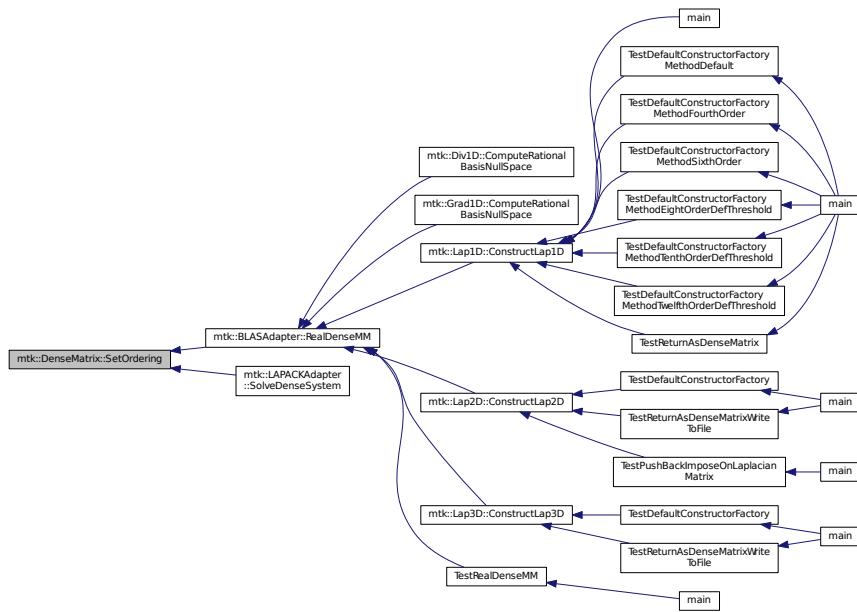
The required value at the specified coordinates.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

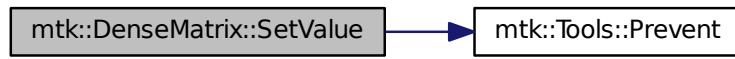
Parameters

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

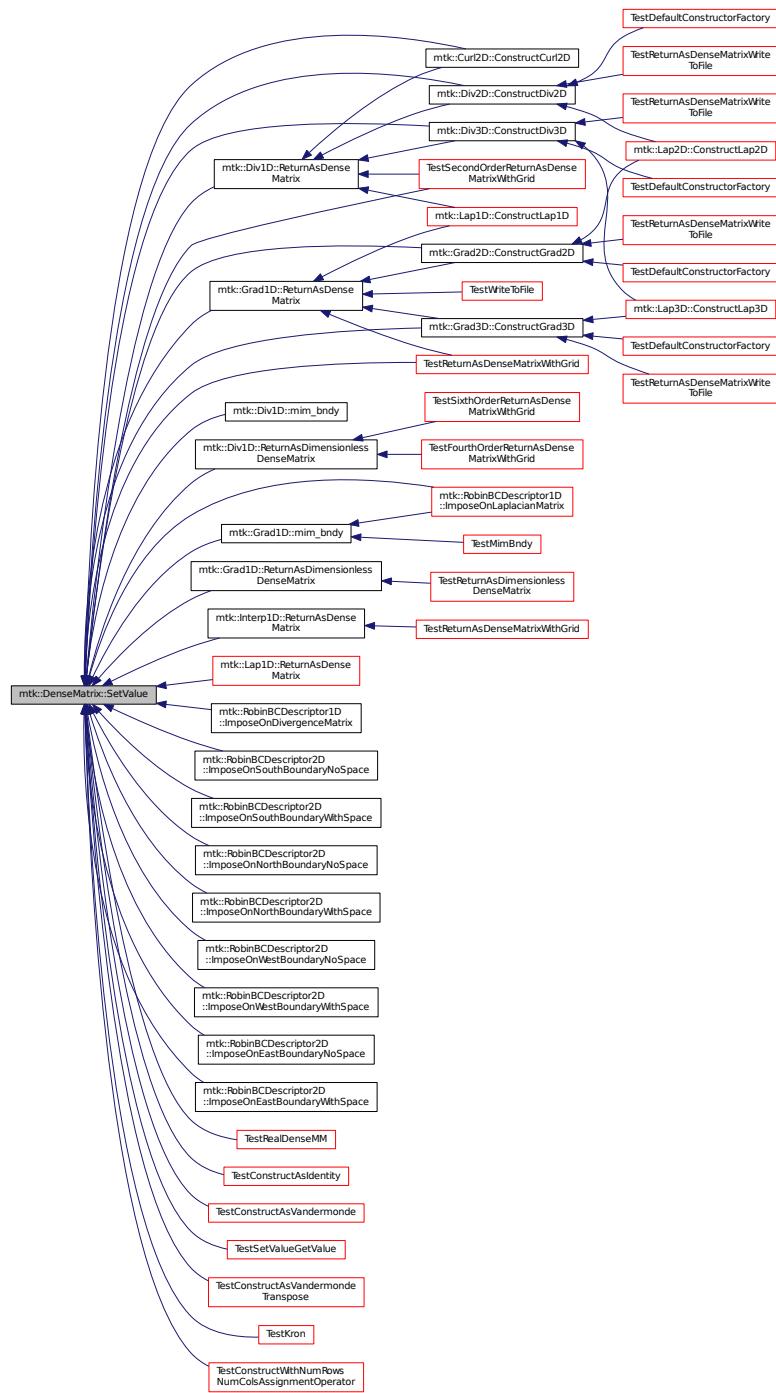
in	val	Row Actual value to be inserted.
----	-----	----------------------------------

Definition at line 404 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:

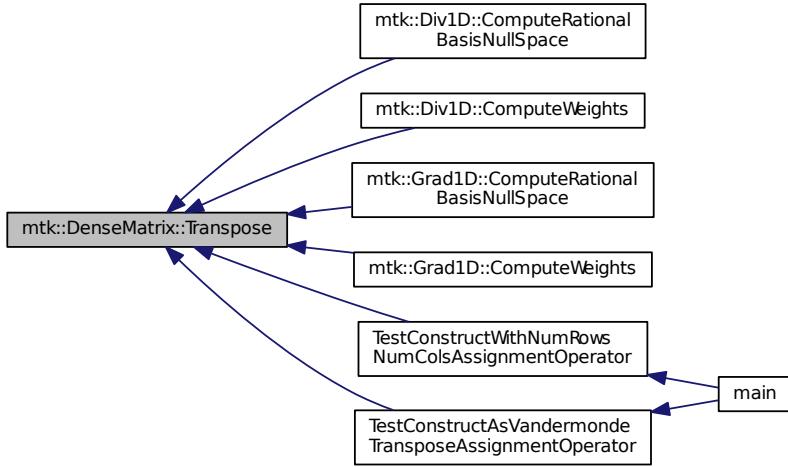


17.3.3.15 void mtk::DenseMatrix::Transpose()

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

Definition at line 417 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



17.3.3.16 bool mtk::DenseMatrix::WriteToFile (const std::string & *filename*) const

Parameters

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

Returns

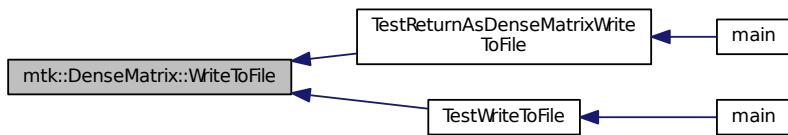
Success of the file writing process.

See also

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

Definition at line 577 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



17.3.4 Friends And Related Function Documentation

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

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

17.3.5 Member Data Documentation

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

Definition at line 320 of file [mtk_dense_matrix.h](#).

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

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

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

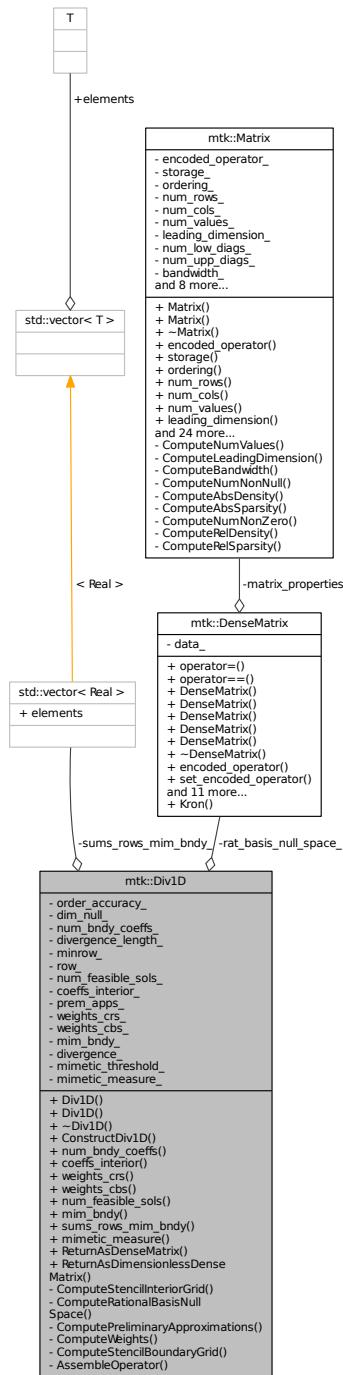
- [include/mtk_dense_matrix.h](#)
- [src/mtk_dense_matrix.cc](#)

17.4 mtk::Div1D Class Reference

Implements a 1D mimetic divergence operator.

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

Collaboration diagram for mtk::Div1D:



Public Member Functions

- [Div1D \(\)](#)

- **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.
- **int num_feasible_sols** () const

Return number of feasible solutions when using the CBSA for weights.
- **DenseMatrix mim_bndy** () const

Return collection of mimetic approximations at the boundary.
- **std::vector< Real > sums_rows_mim_bndy** () const

Return collection of row-sums mimetic approximations at the boundary.
- **Real mimetic_measure** () const

Returns mimetic measure of the operator.
- **DenseMatrix ReturnAsDenseMatrix** (const **UniStgGrid1D** &grid) const

Return the operator as a dense matrix.
- **DenseMatrix ReturnAsDimensionlessDenseMatrix** (int num_cells_x) const

Returns the operator as a dimensionless dense matrix.

Private Member Functions

- **bool ComputeStencilInteriorGrid** (void)

Stage 1 of the CBS Algorithm.
- **bool ComputeRationalBasisNullSpace** (void)

Stage 2.1 of the CBS Algorithm.
- **bool ComputePreliminaryApproximations** (void)

Stage 2.2 of the CBS Algorithm.
- **bool ComputeWeights** (void)

Stage 2.3 of the CBS Algorithm.
- **bool ComputeStencilBoundaryGrid** (void)

Stage 2.4 of the CBS Algorithm.
- **bool AssembleOperator** (void)

Stage 3 of the CBS Algorithm.

Private Attributes

- int `order_accuracy_`
Order of numerical accuracy of the operator.
- int `dim_null_`
Dim. null-space for boundary approximations.
- int `num_bndy_coeffs_`
Req. coeffs. per bndy pt. uni. order accuracy.
- int `divergence_length_`
Length of the output array.
- int `minrow_`
Row from the optimizer with the minimum rel. nor.
- int `row_`
Row currently processed by the optimizer.
- int `num_feasible_sols_`
Number of feasible solutions for weights.
- `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.
- `Real mimetic_measure_`
< Mimetic measure.
- `std::vector< Real > sums_rows_mim_bndy_`
Sum of each mimetic boundary row.

Friends

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

17.4.1 Detailed Description

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

Definition at line 83 of file [mtk_div_1d.h](#).

17.4.2 Constructor & Destructor Documentation

17.4.2.1 mtk::Div1D::Div1D()

Definition at line 134 of file [mtk_div_1d.cc](#).

17.4.2.2 mtk::Div1D::Div1D(const Div1D & div)

Parameters

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

Definition at line 152 of file [mtk_div_1d.cc](#).

17.4.2.3 mtk::Div1D::~Div1D()

Definition at line 170 of file [mtk_div_1d.cc](#).

17.4.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

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

Definition at line 1492 of file [mtk_div_1d.cc](#).

17.4.3.2 mtk::Real * mtk::Div1D::coeffs_interior() const

Returns

Coefficients for the interior of the grid.

Definition at line 335 of file [mtk_div_1d.cc](#).

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

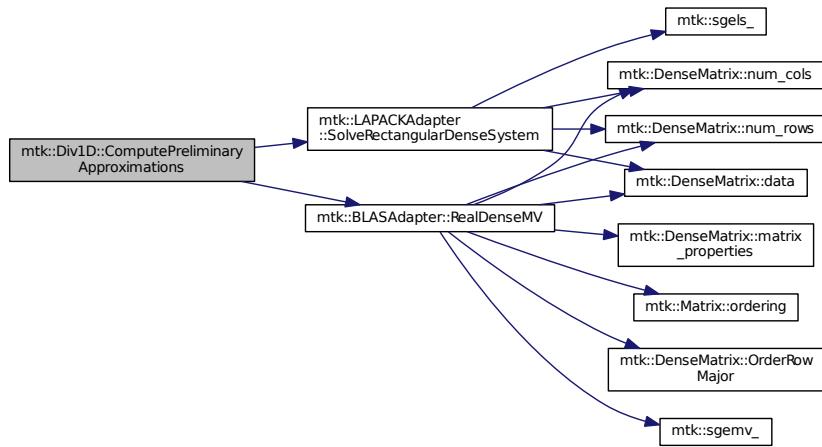
Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.
2. Compute the `dim_null` near-the-boundary columns of the `pi` matrix.
3. Create the Vandermonde matrix for this iteration.
4. New order-selector vector (gets re-written with LAPACK solutions).

5. Solving $TT^*rr = ob$ yields the columns rr of the KK matrix.
6. Scale the KK matrix to make it a rational basis for null-space.
7. Extract the last dim_null values of the pre-scaled ob .
8. Once we posses the bottom elements, we proceed with the scaling.

Definition at line 773 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



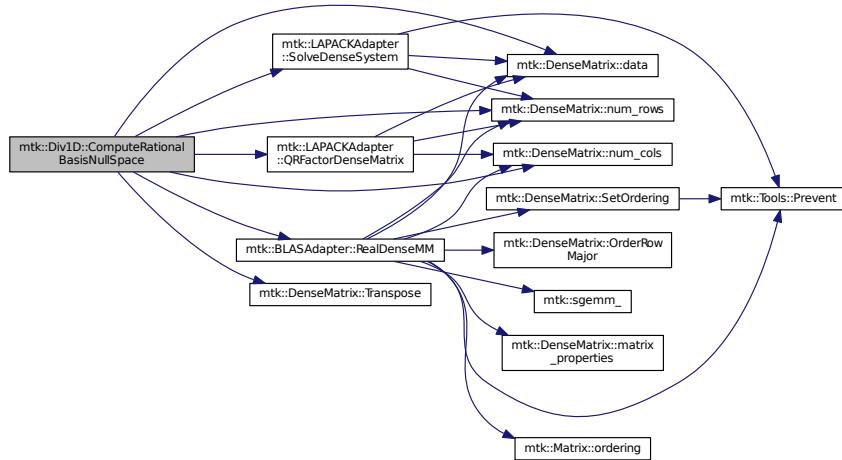
17.4.3.4 bool mtk::Div1D::ComputeRationalBasisNullSpace (void) [private]

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

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

Definition at line 597 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



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

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.
2. Compute alpha values.
3. Compute the mimetic boundary approximations.
4. Compute the row-wise sum to double-check the operator is mimetic.

Definition at line 1370 of file [mtk_div_1d.cc](#).

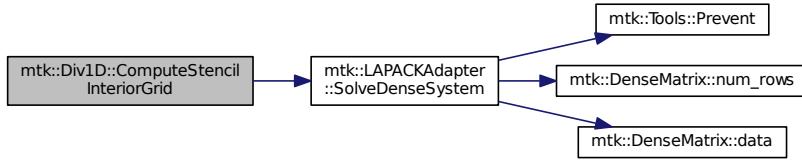
17.4.3.6 bool mtk::Div1D::ComputeStencilInteriorGrid (void) [private]

Compute the stencil approximating the interior of the staggered grid.

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

Definition at line 496 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



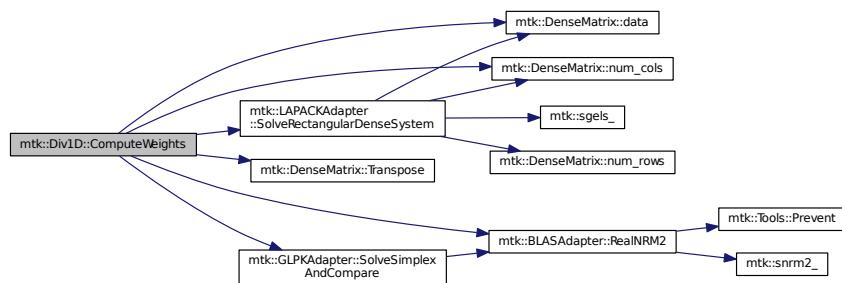
17.4.3.7 bool mtk::Div1D::ComputeWeights (void) [private]

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

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

Definition at line 993 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



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

Returns

Success of the construction.

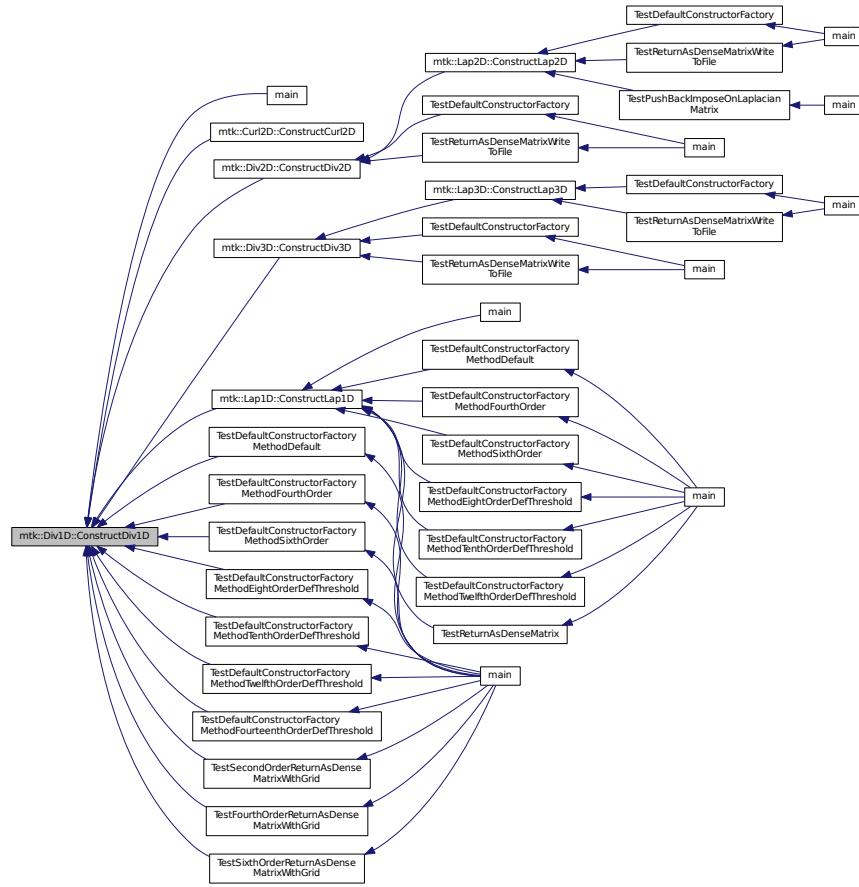
1. Compute stencil for the interior cells.
2. Compute a rational basis for the null-space for the first matrix.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 191 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Returns

Collection of mimetic approximations at the boundary.

Definition at line 355 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



17.4.3.10 mtk::Real mtk::Div1D::mimetic_measure () const**Returns**

Real number which is the mimetic measure of the operator.

Definition at line 375 of file [mtk_div_1d.cc](#).

17.4.3.11 int mtk::Div1D::num_bndy_coeffs () const**Returns**

How many coefficients are approximating at the boundary.

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

17.4.3.12 int mtk::Div1D::num_feasible_sols () const**Returns**

Return number of feasible solutions when using the CBSA for weights.

Definition at line 350 of file [mtk_div_1d.cc](#).

Here is the caller graph for this function:

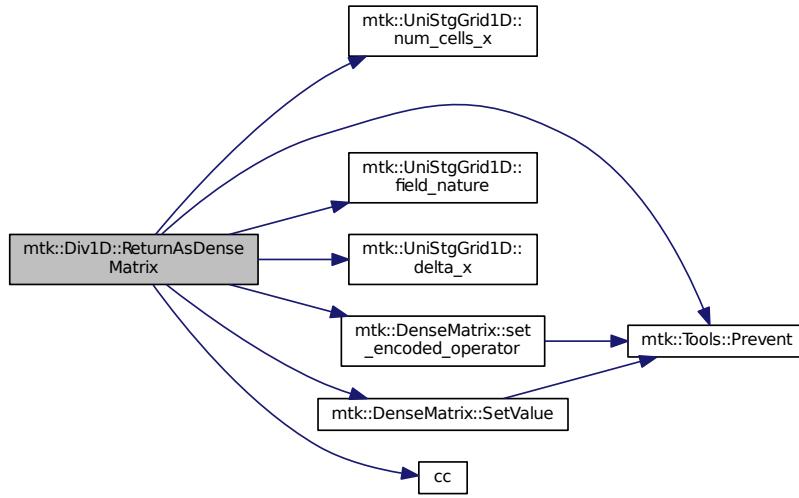
**17.4.3.13 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid) const****Returns**

The operator as a dense matrix.

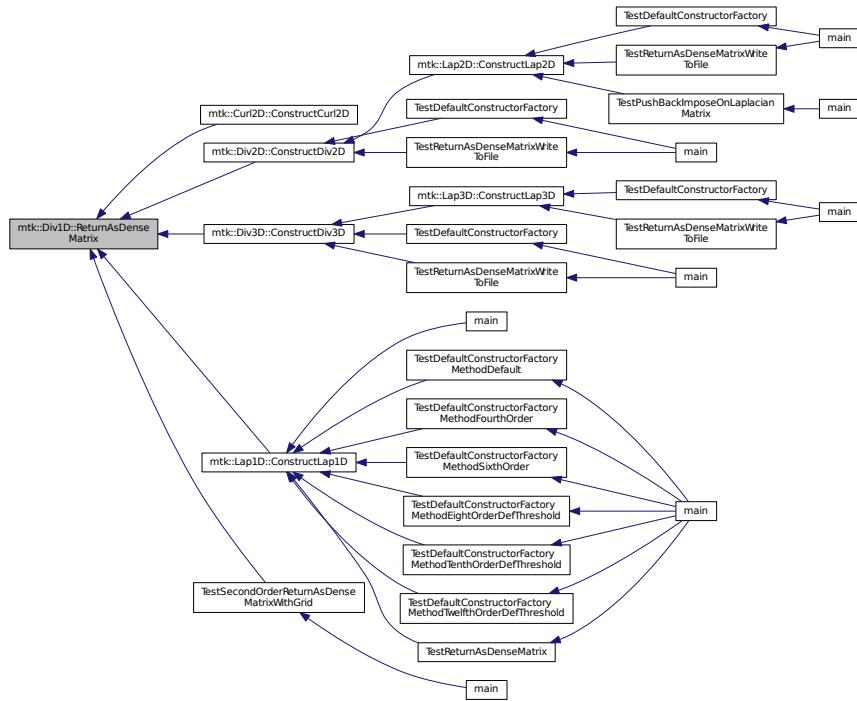
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 380 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.4.3.14 mtk::DenseMatrix mtk::Div1D::ReturnAsDimensionlessDenseMatrix (int num_cells_x) const

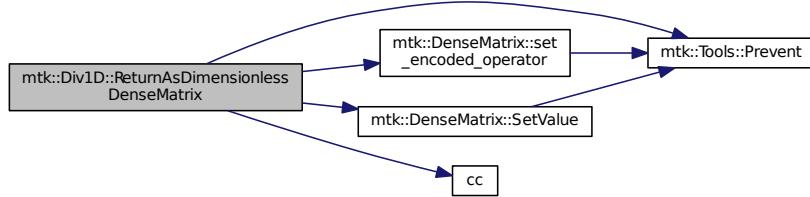
Returns

The operator as a dimensionless dense matrix.

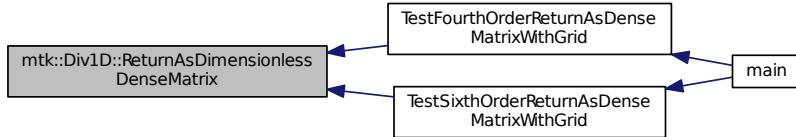
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 440 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.4.3.15 std::vector< mtk::Real > mtk::Div1D::sums_rows_mim_bndy () const

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 370 of file [mtk_div_1d.cc](#).

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

Returns

Collection of weights as computed by the CBSA.

Definition at line 345 of file [mtk_div_1d.cc](#).

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

Returns

Collection of weights as computed by the CRSA.

Definition at line 340 of file [mtk_div_1d.cc](#).

17.4.4 Friends And Related Function Documentation

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

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

Definition at line 82 of file [mtk_div_1d.cc](#).

17.4.5 Member Data Documentation

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

Definition at line 239 of file [mtk_div_1d.h](#).

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

Definition at line 230 of file [mtk_div_1d.h](#).

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

Definition at line 244 of file [mtk_div_1d.h](#).

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

Definition at line 232 of file [mtk_div_1d.h](#).

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

Definition at line 243 of file [mtk_div_1d.h](#).

17.4.5.6 **Real mtk::Div1D::mimetic_measure_ [private]**

Definition at line 247 of file [mtk_div_1d.h](#).

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

Definition at line 246 of file [mtk_div_1d.h](#).

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

Definition at line 233 of file [mtk_div_1d.h](#).

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

Definition at line 231 of file [mtk_div_1d.h](#).

17.4.5.10 **int** `mtk::Div1D::num_feasible_sols_` [private]

Definition at line 235 of file [mtk_div_1d.h](#).

17.4.5.11 **int** `mtk::Div1D::order_accuracy_` [private]

Definition at line 229 of file [mtk_div_1d.h](#).

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

Definition at line 240 of file [mtk_div_1d.h](#).

17.4.5.13 **DenseMatrix** `mtk::Div1D::rat_basis_null_space_` [private]

Definition at line 237 of file [mtk_div_1d.h](#).

17.4.5.14 **int** `mtk::Div1D::row_` [private]

Definition at line 234 of file [mtk_div_1d.h](#).

17.4.5.15 **std::vector<Real>** `mtk::Div1D::sums_rows_mim_bndy_` [private]

Definition at line 249 of file [mtk_div_1d.h](#).

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

Definition at line 242 of file [mtk_div_1d.h](#).

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

Definition at line 241 of file [mtk_div_1d.h](#).

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

- [include/mtk_div_1d.h](#)

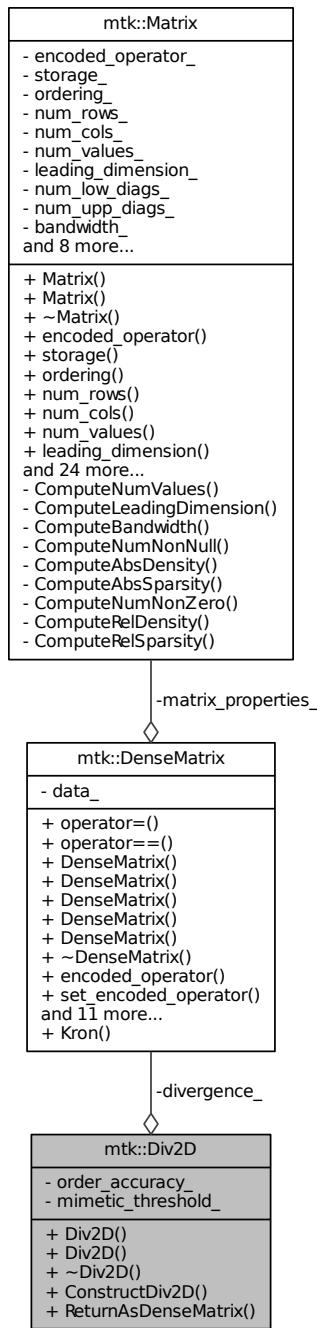
- [src/mtk_div_1d.cc](#)

17.5 mtk::Div2D Class Reference

Implements a 2D mimetic divergence operator.

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

Collaboration diagram for mtk::Div2D:



Public Member Functions

- [Div2D \(\)](#)

- *Default constructor.*
- `Div2D (const Div2D &div)`
- Copy constructor.*
- `~Div2D ()`
- Destructor.*
- `bool ConstructDiv2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`
- Factory method implementing the CBS Algorithm to build operator.*
- `DenseMatrix ReturnAsDenseMatrix () const`
- Return the operator as a dense matrix.*

Private Attributes

- `DenseMatrix divergence_`
Actual operator.
- `int order_accuracy_`
Order of accuracy.
- `Real mimetic_threshold_`
Mimetic Threshold.

17.5.1 Detailed Description

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

Definition at line 76 of file [mtk_div_2d.h](#).

17.5.2 Constructor & Destructor Documentation

17.5.2.1 `mtk::Div2D::Div2D()`

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

17.5.2.2 `mtk::Div2D::Div2D (const Div2D & div)`

Parameters

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

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

17.5.2.3 `mtk::Div2D::~Div2D()`

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

17.5.3 Member Function Documentation

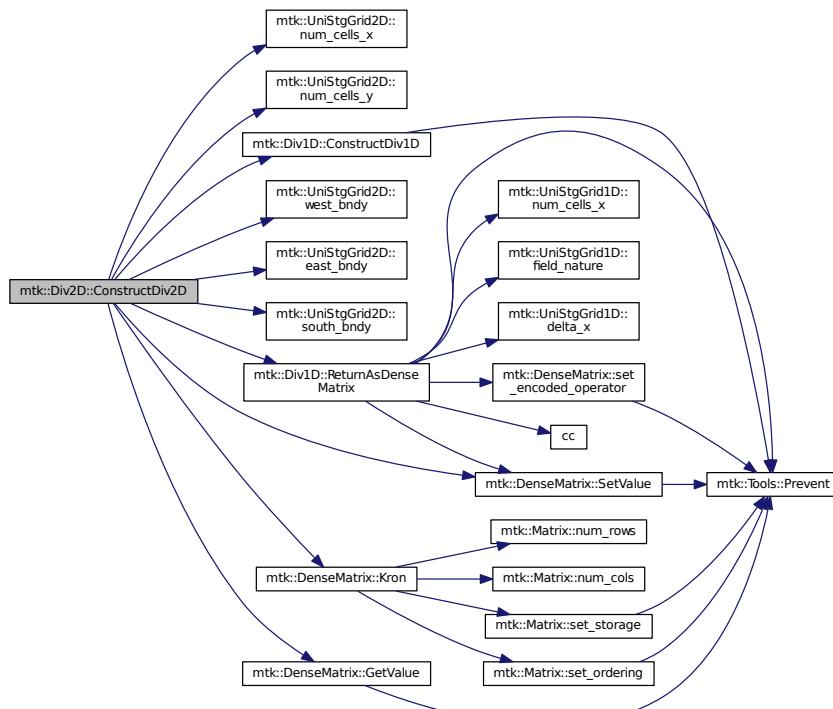
17.5.3.1 `bool mtk::Div2D::ConstructDiv2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

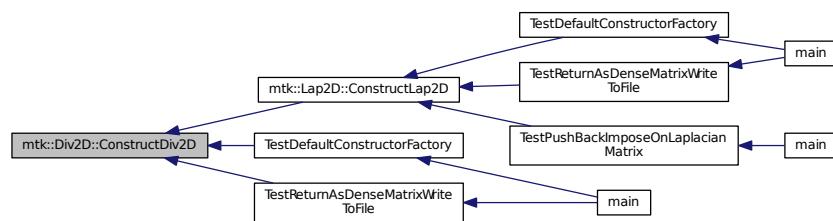
Success of the construction.

Definition at line 79 of file [mtk_div_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



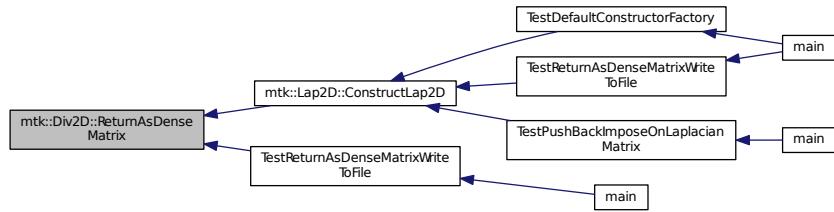
17.5.3.2 `mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() const`

Returns

The operator as a dense matrix.

Definition at line 147 of file [mtk_div_2d.cc](#).

Here is the caller graph for this function:



17.5.4 Member Data Documentation

17.5.4.1 `DenseMatrix mtk::Div2D::divergence_ [private]`

Definition at line 112 of file [mtk_div_2d.h](#).

17.5.4.2 `Real mtk::Div2D::mimetic_threshold_ [private]`

Definition at line 116 of file [mtk_div_2d.h](#).

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

Definition at line 114 of file [mtk_div_2d.h](#).

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

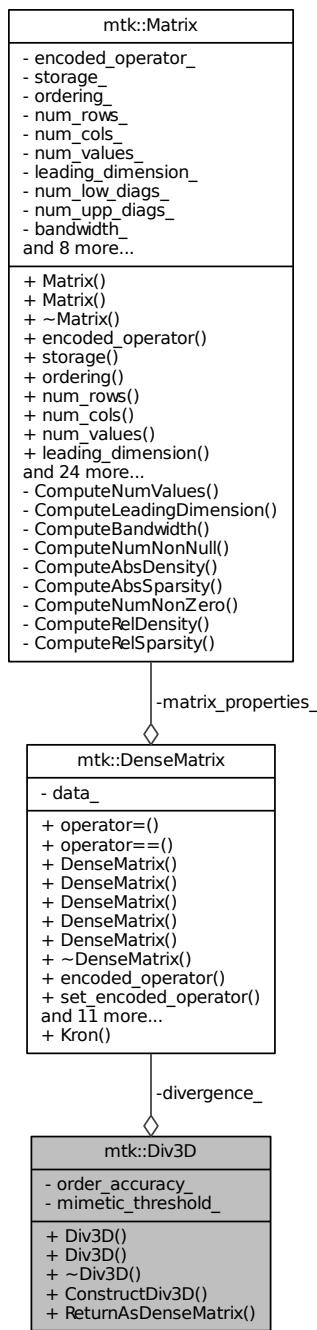
- [include/mtk_div_2d.h](#)
- [src/mtk_div_2d.cc](#)

17.6 `mtk::Div3D Class Reference`

Implements a 3D mimetic divergence operator.

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

Collaboration diagram for mtk::Div3D:



Public Member Functions

- [Div3D \(\)](#)

- *Default constructor.*
- `Div3D (const Div3D &div)`
- Copy constructor.*
- `~Div3D ()`
- Destructor.*
- `bool ConstructDiv3D (const UniStgGrid3D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`
- Factory method implementing the CBS Algorithm to build operator.*
- `DenseMatrix ReturnAsDenseMatrix () const`
- Return the operator as a dense matrix.*

Private Attributes

- `DenseMatrix divergence_`
Actual operator.
- `int order_accuracy_`
Order of accuracy.
- `Real mimetic_threshold_`
Mimetic Threshold.

17.6.1 Detailed Description

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

Definition at line 76 of file [mtk_div_3d.h](#).

17.6.2 Constructor & Destructor Documentation

17.6.2.1 `mtk::Div3D::Div3D()`

Definition at line 67 of file [mtk_div_3d.cc](#).

17.6.2.2 `mtk::Div3D::Div3D (const Div3D & div)`

Parameters

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

Definition at line 71 of file [mtk_div_3d.cc](#).

17.6.2.3 `mtk::Div3D::~Div3D()`

Definition at line 75 of file [mtk_div_3d.cc](#).

17.6.3 Member Function Documentation

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

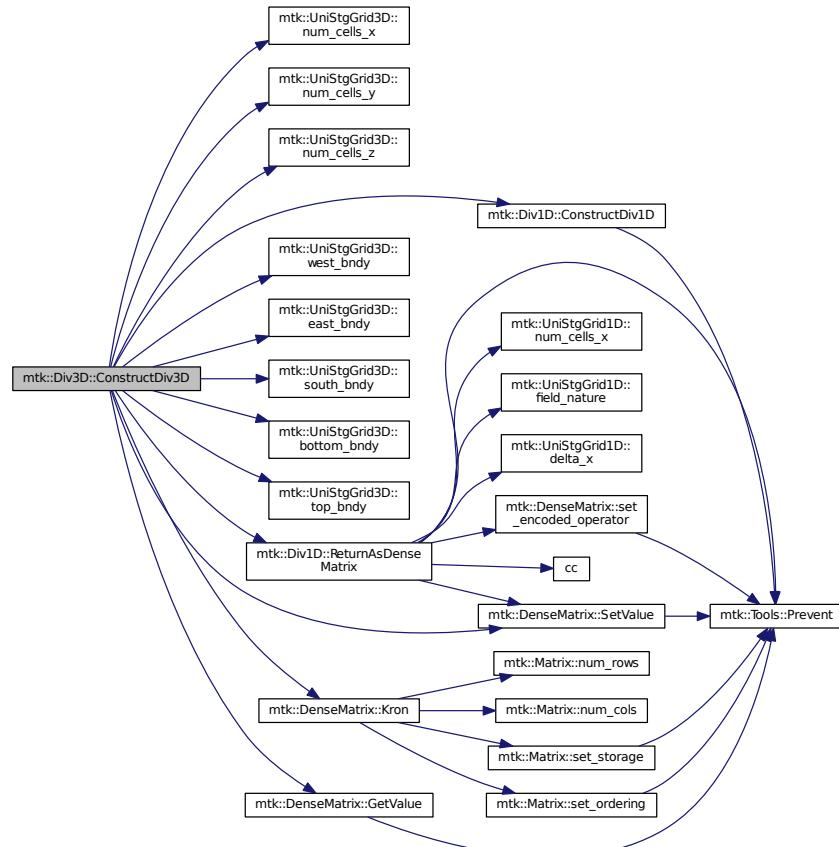
Returns

Success of the construction.

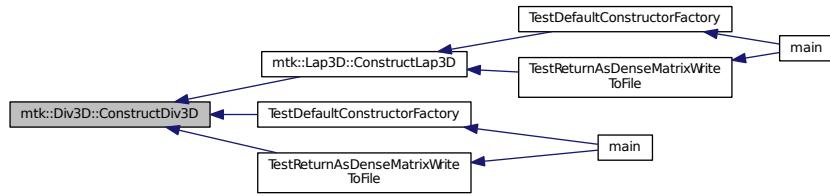
1. Build preliminary staggering through the x direction.
2. Build preliminary staggering through the y direction.
3. Build preliminary staggering through the z direction.
4. Actual operator: $\text{DD_xyz} = [\text{dx dy dz}]$.

Definition at line 77 of file [mtk_div_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



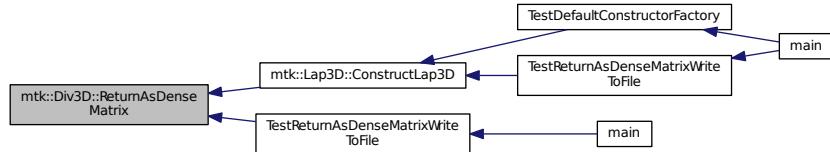
17.6.3.2 mtk::DenseMatrix mtk::Div3D::ReturnAsDenseMatrix () const

Returns

The operator as a dense matrix.

Definition at line 186 of file [mtk_div_3d.cc](#).

Here is the caller graph for this function:



17.6.4 Member Data Documentation

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

Definition at line 112 of file [mtk_div_3d.h](#).

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

Definition at line 116 of file [mtk_div_3d.h](#).

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

Definition at line 114 of file [mtk_div_3d.h](#).

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

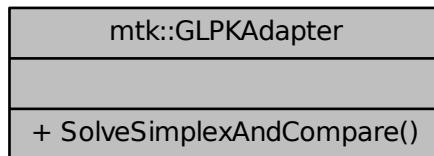
- [include/mtk_div_3d.h](#)
- [src/mtk_div_3d.cc](#)

17.7 mtk::GLPKAdapter Class Reference

Adapter class for the GLPK API.

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

Collaboration diagram for mtk::GLPKAdapter:



Static Public Member Functions

- static `mtk::Real SolveSimplexAndCompare (mtk::Real *AA, int nrows, int ncols, int kk, mtk::Real *hh, mtk::Real *qq, int robjective, mtk::Real mimetic_threshold, int copy) noexcept`
Solves a CLO problem and compares the solution to a reference solution.

17.7.1 Detailed Description

This class contains a collection of static member functions, that possess 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/>

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

17.7.2 Member Function Documentation

- 17.7.2.1 `mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare (mtk::Real * AA, int nrows, int ncols, int kk, mtk::Real * hh, mtk::Real * qq, int robjective, mtk::Real mimetic_threshold, int copy) [static], [noexcept]`

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>AA</i>	Given matrix.
in	<i>nrows</i>	Number of rows of the matrix.
in	<i>ncols</i>	Number of columns of the matrix.
in	<i>kk</i>	Length of the RHS vector of constraints.
in	<i>hh</i>	RHS vector of constraints.
in,out	<i>qq</i>	Output decision vector.
in	<i>robjective</i>	Row of the system to be chosen as objective function.
in	<i>mimetic_</i> ← <i>threshold</i>	Mimetic threshold.
in	<i>copy</i>	Should we actually copy the results to the output?

Returns

Relative error computed between attained solution and provided ref.

Warning

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

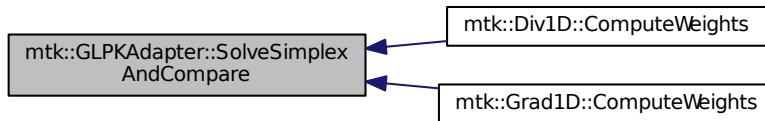
1. Memory allocation.
2. Fill the problem.
3. Copy the row to the vector objective.
4. Forming the RHS.
5. Setting up the objective function.
6. Setting up constraints.
7. Copy the matrix minus the row objective to the glpk problem.
8. Solve problem.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

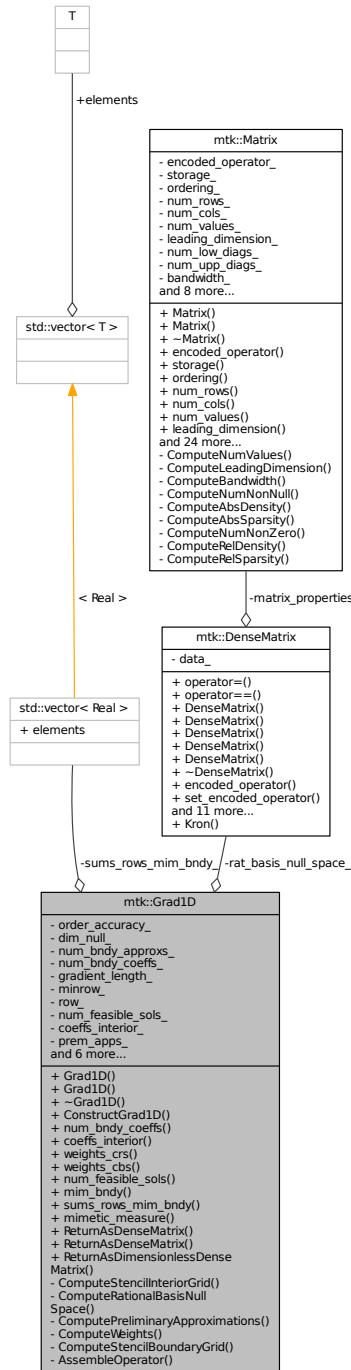
- include/[mtk_glpk_adapter.h](#)
- src/[mtk_glpk_adapter.cc](#)

17.8 mtk::Grad1D Class Reference

Implements a 1D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad1D:



Public Member Functions

- [Grad1D \(\)](#)

- **Grad1D** (const **Grad1D** &grad)
 - Default constructor.*
- **~Grad1D** ()
 - Copy constructor.*
 - 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.*
- int **num_feasible_sols** () const
 - Return number of feasible solutions when using the CBSA for weights.*
- DenseMatrix **mim_bndy** () const
 - Return collection of mimetic approximations at the boundary.*
- std::vector< Real > **sums_rows_mim_bndy** () const
 - Return collection of row-sums mimetic approximations at the boundary.*
- Real **mimetic_measure** () const
 - Returns mimetic measure of the operator.*
- DenseMatrix **ReturnAsDenseMatrix** (Real west, Real east, int num_cells_x) const
 - Returns the operator as a dense matrix.*
- DenseMatrix **ReturnAsDenseMatrix** (const UniStgGrid1D &grid) const
 - Returns the operator as a dense matrix.*
- DenseMatrix **ReturnAsDimensionlessDenseMatrix** (int num_cells_x) const
 - Returns the operator as a dimensionless dense matrix.*

Private Member Functions

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

Private Attributes

- int `order_accuracy_`
Order of numerical accuracy of the operator.
- int `dim_null_`
Dim. null-space for boundary approximations.
- int `num_bndy_approx_`
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.
- int `num_feasible_sols_`
Number of feasible solutions for weights.
- `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.
- `Real mimetic_measure_`
< Mimetic measure.
- `std::vector< Real > sums_rows_mim_bndy_`
Sum of each mimetic boundary row.

Friends

- `std::ostream & operator<< (std::ostream & stream, Grad1D & in)`
Output stream operator for printing.

17.8.1 Detailed Description

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

Definition at line 84 of file [mtk_grad_1d.h](#).

17.8.2 Constructor & Destructor Documentation

17.8.2.1 mtk::Grad1D::Grad1D()

Definition at line 146 of file [mtk_grad_1d.cc](#).

17.8.2.2 mtk::Grad1D::Grad1D(const Grad1D & grad)

Parameters

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

Definition at line 165 of file [mtk_grad_1d.cc](#).

17.8.2.3 mtk::Grad1D::~Grad1D()

Definition at line 184 of file [mtk_grad_1d.cc](#).

17.8.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

1. The first entry of the array will contain the order of accuracy.
2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.
3. The third entry will contain the collection of weights.
4. The next $\dim_{\text{null}} + 1$ entries will contain the collections of approximating coefficients for the west boundary of the grid.

Definition at line 1616 of file [mtk_grad_1d.cc](#).

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

Returns

Coefficients for the interior of the grid.

Definition at line 349 of file [mtk_grad_1d.cc](#).

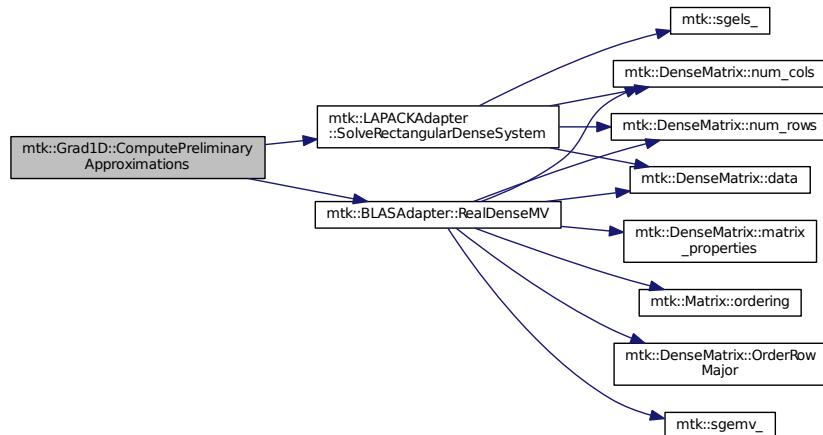
17.8.3.3 bool mtk::Grad1D::ComputePreliminaryApproximations (void) [private]

Compute the set of preliminary approximations on the boundary neighborhood.

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

Definition at line 877 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



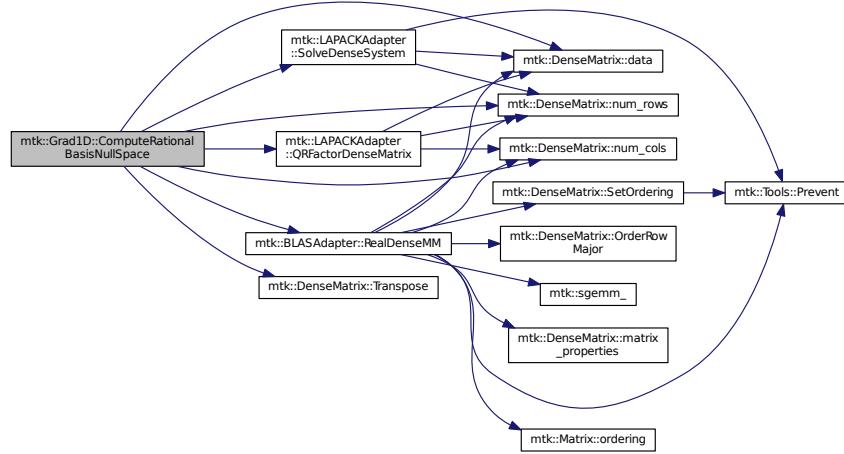
17.8.3.4 bool mtk::Grad1D::ComputeRationalBasisNullSpace (void) [private]

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

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

Definition at line 694 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



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

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.
2. Compute alpha values.
3. Compute the mimetic boundary approximations.
4. Compute the row-wise sum to double-check that the operator is mimetic.

Definition at line 1485 of file [mtk_grad_1d.cc](#).

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

Compute the stencil approximating the interior of the staggered grid.

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

Definition at line 597 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



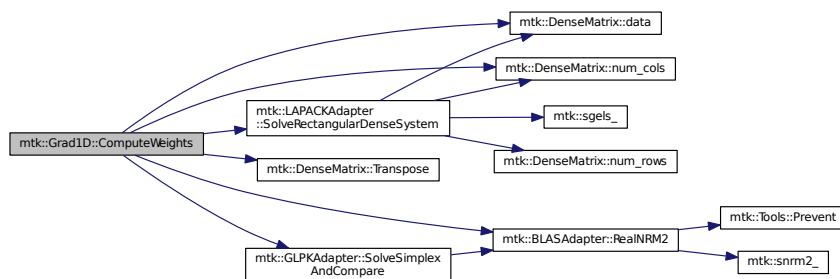
17.8.3.7 bool mtk::Grad1D::ComputeWeights (void) [private]

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

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

Definition at line 1098 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



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

Returns

Success of the solution.

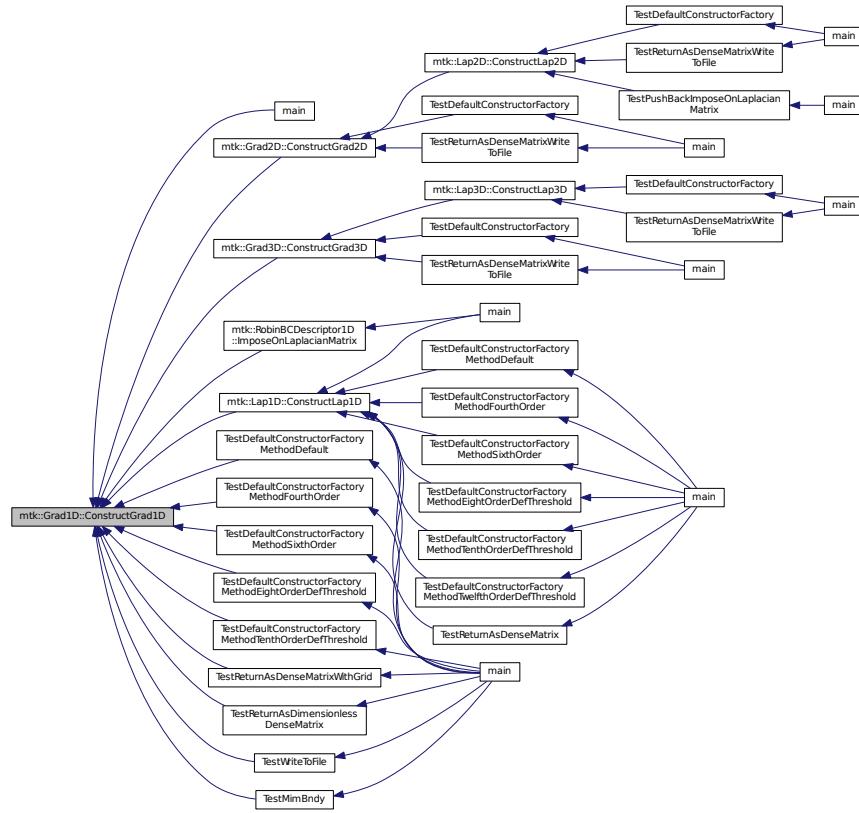
1. Compute stencil for the interior cells.
2. Compute a rational null-space from the first matrix transposed.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 205 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Returns

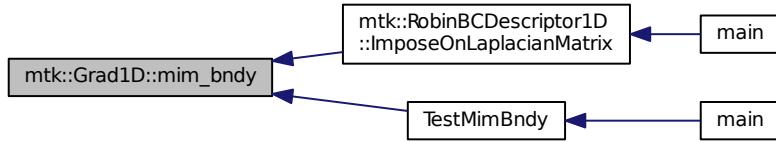
Collection of mimetic approximations at the boundary.

Definition at line 369 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**17.8.3.10 mtk::Real mtk::Grad1D::mimetic_measure () const****Returns**

Real number which is the mimetic measure of the operator.

Definition at line 389 of file [mtk_grad_1d.cc](#).

17.8.3.11 int mtk::Grad1D::num_bndy_coeffs () const**Returns**

How many coefficients are approximating at the boundary.

Definition at line 344 of file [mtk_grad_1d.cc](#).

17.8.3.12 int mtk::Grad1D::num_feasible_sols () const

Returns

Return number of feasible solutions when using the CBSA for weights.

Definition at line 364 of file [mtk_grad_1d.cc](#).

Here is the caller graph for this function:



17.8.3.13 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (mtk::Real west, mtk::Real east, int num_cells_x) const

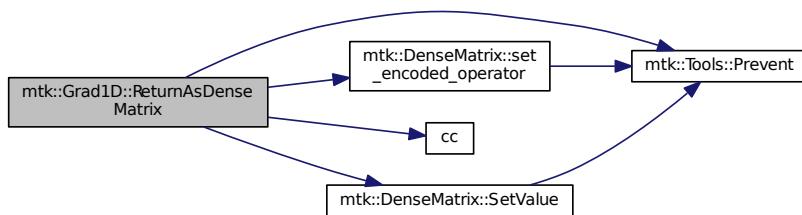
Returns

The operator as a dense matrix.

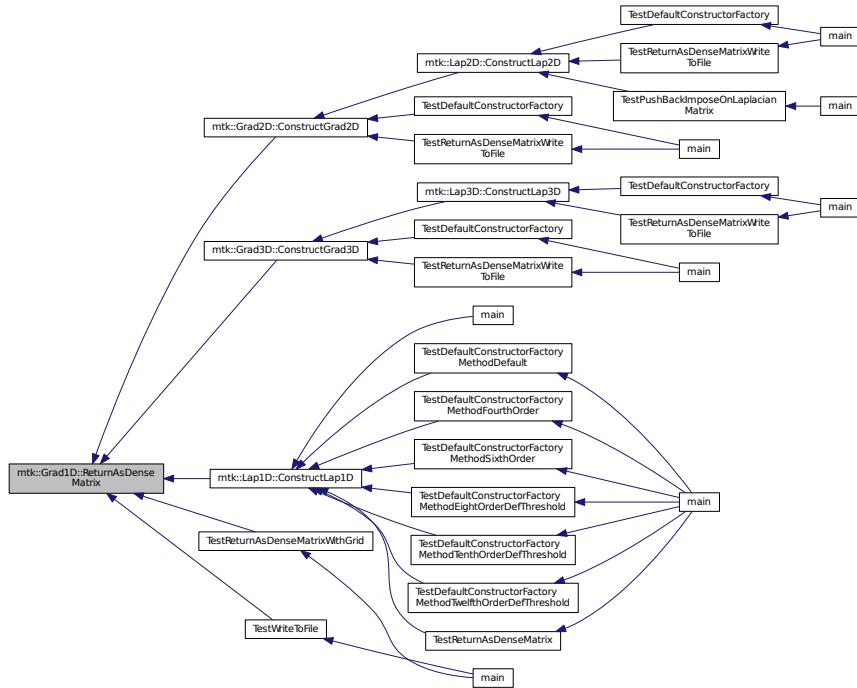
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 394 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.8.3.14 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid) const

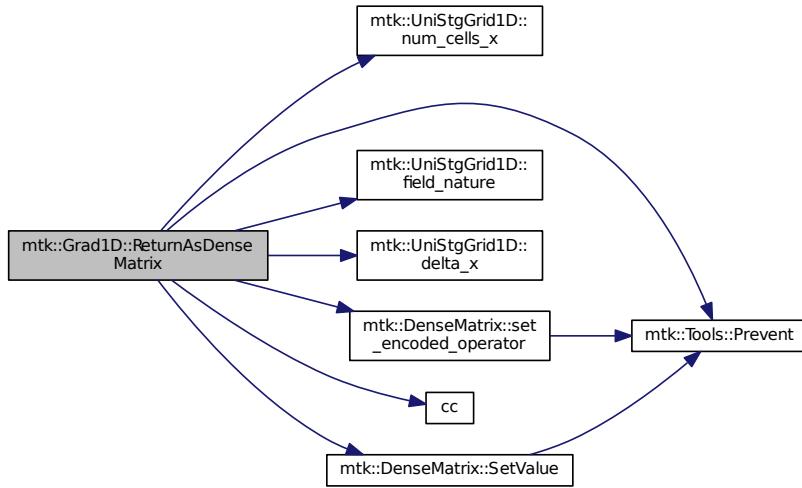
Returns

The operator as a dense matrix.

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

Definition at line 465 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



17.8.3.15 `mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix (int num_cells_x) const`

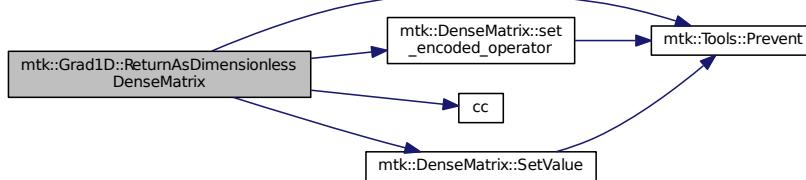
Returns

The operator as a dimensionless dense matrix.

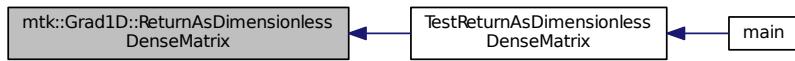
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 533 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.8.3.16 std::vector< mtk::Real > mtk::Grad1D::sums_rows_mim_bndy () const

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 384 of file [mtk_grad_1d.cc](#).

17.8.3.17 mtk::Real * mtk::Grad1D::weights_cbs (void) const

Returns

Collection of weights as computed by the CBSA.

Definition at line 359 of file [mtk_grad_1d.cc](#).

17.8.3.18 mtk::Real * mtk::Grad1D::weights_crs (void) const

Returns

Success of the solution.

Definition at line 354 of file [mtk_grad_1d.cc](#).

17.8.4 Friends And Related Function Documentation

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

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

Definition at line 82 of file [mtk_grad_1d.cc](#).

17.8.5 Member Data Documentation

17.8.5.1 Real* mtk::Grad1D::coeffs_interior_ [private]

Definition at line 248 of file [mtk_grad_1d.h](#).

17.8.5.2 `int mtk::Grad1D::dim_null_ [private]`

Definition at line 238 of file [mtk_grad_1d.h](#).

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

Definition at line 253 of file [mtk_grad_1d.h](#).

17.8.5.4 `int mtk::Grad1D::gradient_length_ [private]`

Definition at line 241 of file [mtk_grad_1d.h](#).

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

Definition at line 252 of file [mtk_grad_1d.h](#).

17.8.5.6 `Real mtk::Grad1D::mimetic_measure_ [private]`

Definition at line 256 of file [mtk_grad_1d.h](#).

17.8.5.7 `Real mtk::Grad1D::mimetic_threshold_ [private]`

Definition at line 255 of file [mtk_grad_1d.h](#).

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

Definition at line 242 of file [mtk_grad_1d.h](#).

17.8.5.9 `int mtk::Grad1D::num_bndy_approx_ [private]`

Definition at line 239 of file [mtk_grad_1d.h](#).

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

Definition at line 240 of file [mtk_grad_1d.h](#).

17.8.5.11 `int mtk::Grad1D::num_feasible_sols_ [private]`

Definition at line 244 of file [mtk_grad_1d.h](#).

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

Definition at line 237 of file [mtk_grad_1d.h](#).

17.8.5.13 **Real* mtk::Grad1D::prem_apps_ [private]**

Definition at line 249 of file [mtk_grad_1d.h](#).

17.8.5.14 **DenseMatrix mtk::Grad1D::rat_basis_null_space_ [private]**

Definition at line 246 of file [mtk_grad_1d.h](#).

17.8.5.15 **int mtk::Grad1D::row_ [private]**

Definition at line 243 of file [mtk_grad_1d.h](#).

17.8.5.16 **std::vector<Real> mtk::Grad1D::sums_rows_mim_bndy_ [private]**

Definition at line 258 of file [mtk_grad_1d.h](#).

17.8.5.17 **Real* mtk::Grad1D::weights_cbs_ [private]**

Definition at line 251 of file [mtk_grad_1d.h](#).

17.8.5.18 **Real* mtk::Grad1D::weights_crs_ [private]**

Definition at line 250 of file [mtk_grad_1d.h](#).

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

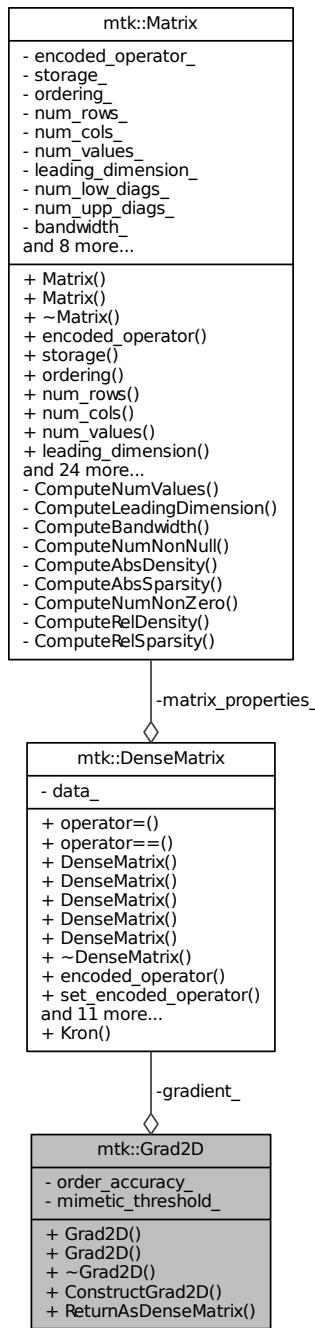
- [include/mtk_grad_1d.h](#)
- [src/mtk_grad_1d.cc](#)

17.9 mtk::Grad2D Class Reference

Implements a 2D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad2D:



Public Member Functions

- [Grad2D \(\)](#)

Default constructor.

- `Grad2D (const Grad2D &grad)`

Copy constructor.

- `~Grad2D ()`

Destructor.

- `bool ConstructGrad2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`

Factory method implementing the CBS Algorithm to build operator.

- `DenseMatrix ReturnAsDenseMatrix () const`

Return the operator as a dense matrix.

Private Attributes

- `DenseMatrix gradient_`

Actual operator.

- `int order_accuracy_`

Order of accuracy.

- `Real mimetic_threshold_`

Mimetic Threshold.

17.9.1 Detailed Description

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

Definition at line 76 of file [mtk_grad_2d.h](#).

17.9.2 Constructor & Destructor Documentation

17.9.2.1 mtk::Grad2D::Grad2D ()

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

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

Parameters

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

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

17.9.2.3 mtk::Grad2D::~Grad2D ()

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

17.9.3 Member Function Documentation

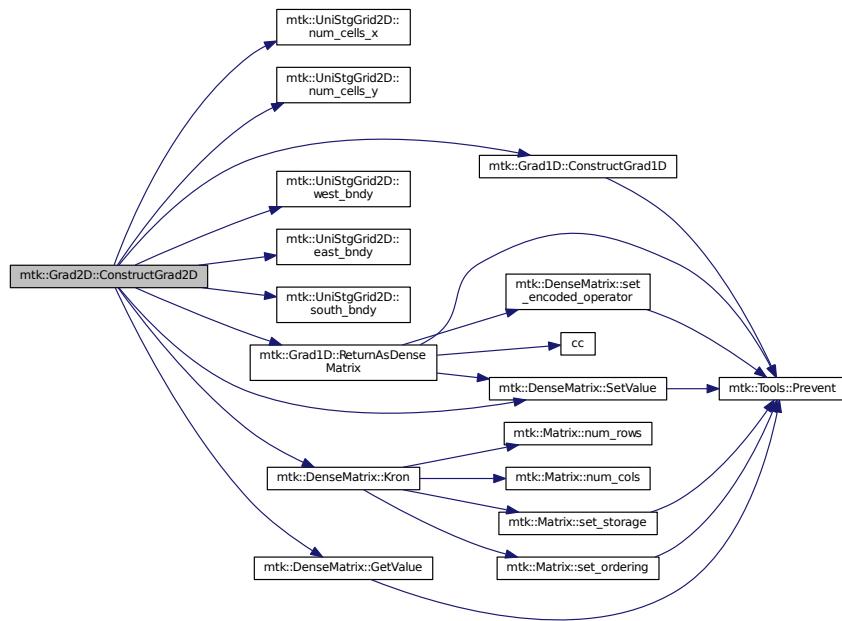
17.9.3.1 `bool mtk::Grad2D::ConstructGrad2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

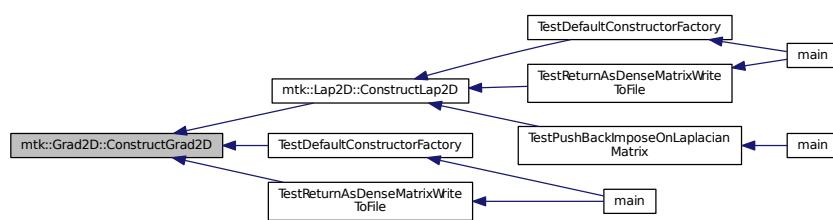
Success of the construction.

Definition at line 77 of file [mtk_grad_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



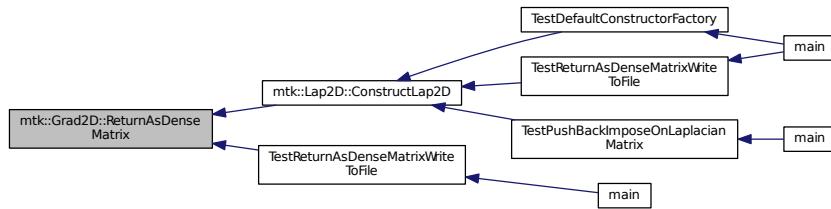
17.9.3.2 `mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix () const`

Returns

The operator as a dense matrix.

Definition at line 145 of file [mtk_grad_2d.cc](#).

Here is the caller graph for this function:



17.9.4 Member Data Documentation

17.9.4.1 DenseMatrix mtk::Grad2D::gradient_ [private]

Definition at line 112 of file [mtk_grad_2d.h](#).

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

Definition at line 116 of file [mtk_grad_2d.h](#).

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

Definition at line 114 of file [mtk_grad_2d.h](#).

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

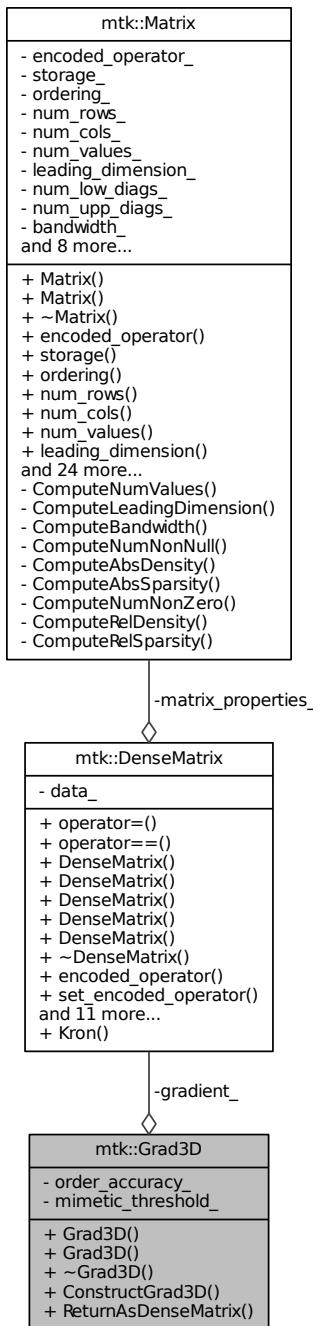
- include/[mtk_grad_2d.h](#)
- src/[mtk_grad_2d.cc](#)

17.10 mtk::Grad3D Class Reference

Implements a 3D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad3D:



Public Member Functions

- [Grad3D \(\)](#)

Default constructor.

- `Grad3D (const Grad3D &grad)`

Copy constructor.

- `~Grad3D ()`

Destructor.

- `bool ConstructGrad3D (const UniStgGrid3D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`

Factory method implementing the CBS Algorithm to build operator.

- `DenseMatrix ReturnAsDenseMatrix () const`

Return the operator as a dense matrix.

Private Attributes

- `DenseMatrix gradient_`

Actual operator.

- `int order_accuracy_`

Order of accuracy.

- `Real mimetic_threshold_`

Mimetic Threshold.

17.10.1 Detailed Description

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

Definition at line 76 of file `mtk_grad_3d.h`.

17.10.2 Constructor & Destructor Documentation

17.10.2.1 mtk::Grad3D::Grad3D()

Definition at line 67 of file `mtk_grad_3d.cc`.

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

Parameters

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

Definition at line 71 of file `mtk_grad_3d.cc`.

17.10.2.3 mtk::Grad3D::~Grad3D()

Definition at line 75 of file `mtk_grad_3d.cc`.

17.10.3 Member Function Documentation

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

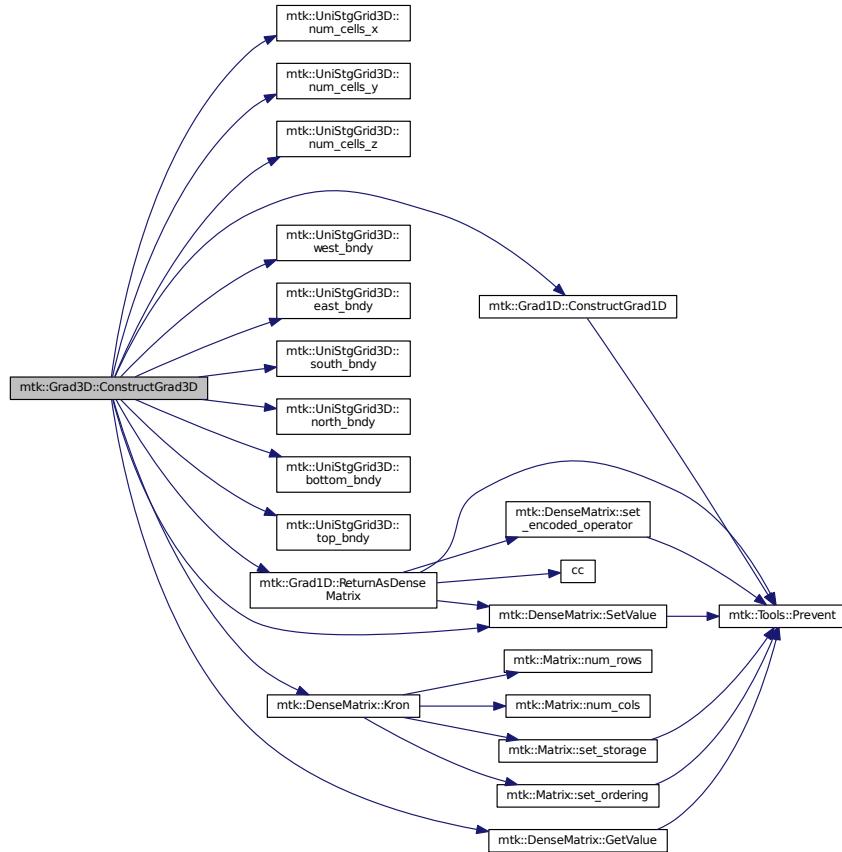
Returns

Success of the construction.

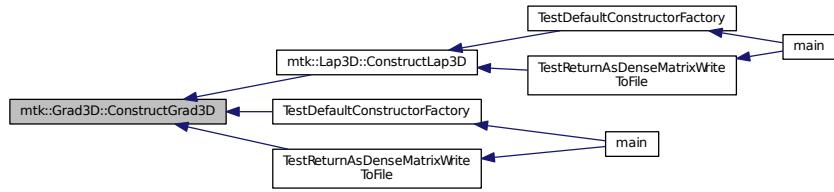
1. Build preliminary staggering through the x direction.
2. Build preliminary staggering through the y direction.
3. Build preliminary staggering through the z direction.
4. Actual operator: $\text{GG_xyz} = [\text{gx}; \text{gy}; \text{gz}]$.

Definition at line 77 of file [mtk_grad_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



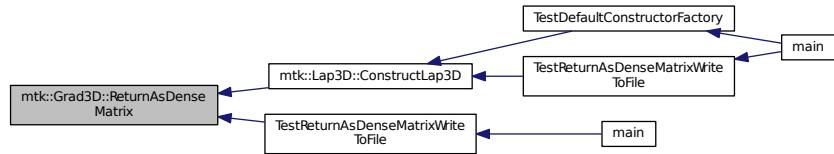
17.10.3.2 mtk::DenseMatrix mtk::Grad3D::ReturnAsDenseMatrix () const

Returns

The operator as a dense matrix.

Definition at line 185 of file [mtk_grad_3d.cc](#).

Here is the caller graph for this function:



17.10.4 Member Data Documentation

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

Definition at line 112 of file [mtk_grad_3d.h](#).

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

Definition at line 116 of file [mtk_grad_3d.h](#).

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

Definition at line 114 of file [mtk_grad_3d.h](#).

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

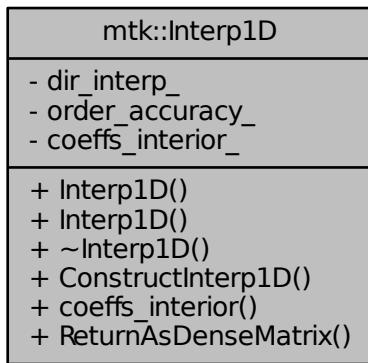
- [include/mtk_grad_3d.h](#)
- [src/mtk_grad_3d.cc](#)

17.11 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

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

Collaboration diagram for mtk::Interp1D:



Public Member Functions

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

Private Attributes

- [DirInterp dir_interp_](#)
Direction of interpolation.
- [int order_accuracy_](#)
Order of numerical accuracy of the operator.
- [Real * coeffs_interior_](#)
Interior stencil.

Friends

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

17.11.1 Detailed Description

This class implements a 1D interpolation operator.

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

17.11.2 Constructor & Destructor Documentation

17.11.2.1 mtk::Interp1D::Interp1D ()

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

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

Parameters

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

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

17.11.2.3 mtk::Interp1D::~Interp1D ()

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

17.11.3 Member Function Documentation

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

Returns

Coefficients for the interior of the grid.

Definition at line 132 of file [mtk_interp_1d.cc](#).

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

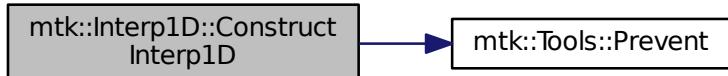
Returns

Success of the solution.

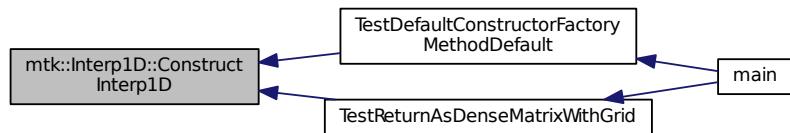
1. Compute stencil for the interior cells.

Definition at line 96 of file [mtk_interp_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

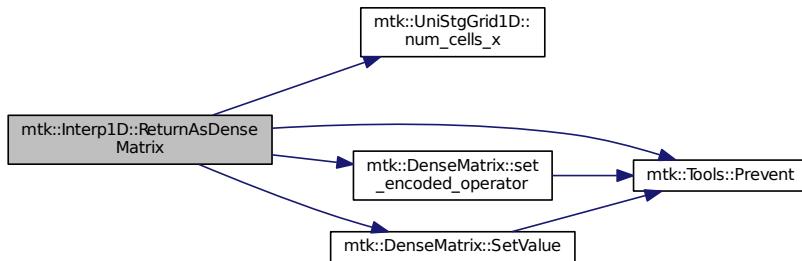
Returns

The operator as a dense matrix.

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

Definition at line 137 of file [mtk_interp_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.11.4 Friends And Related Function Documentation

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

1. Print approximating coefficients for the interior.

Definition at line 66 of file [mtk_interp_1d.cc](#).

17.11.5 Member Data Documentation

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

Definition at line 133 of file [mtk_interp_1d.h](#).

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

Definition at line 129 of file [mtk_interp_1d.h](#).

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

Definition at line 131 of file [mtk_interp_1d.h](#).

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

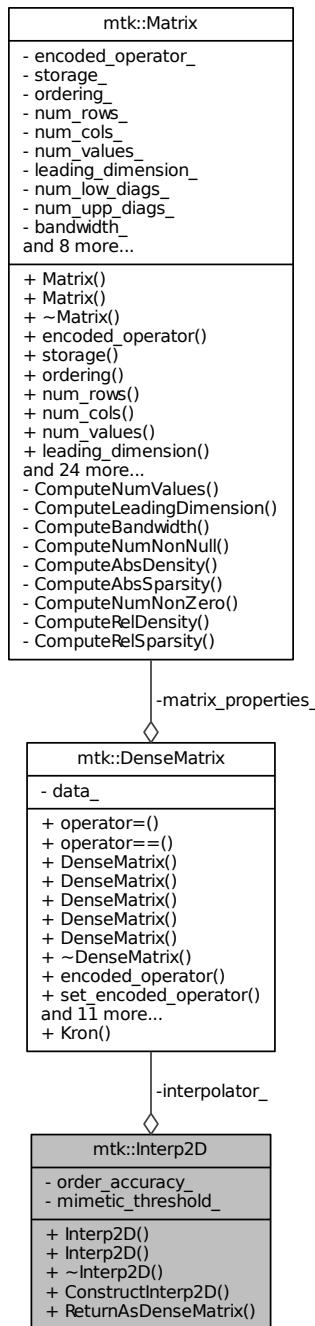
- [include/mtk_interp_1d.h](#)
- [src/mtk_interp_1d.cc](#)

17.12 mtk::Interp2D Class Reference

Implements a 2D interpolation operator.

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

Collaboration diagram for mtk::Interp2D:



Public Member Functions

- [Interp2D \(\)](#)

- *Default constructor.*
- `Interp2D (const Interp2D &interp)`
 - Copy constructor.*
- `~Interp2D ()`
 - Destructor.*
- `DenseMatrix ConstructInterp2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`
 - Factory method implementing the CBS Algorithm to build operator.*
- `DenseMatrix ReturnAsDenseMatrix ()`
 - Return the operator as a dense matrix.*

Private Attributes

- `DenseMatrix interpolator_`
 - Actual operator.*
- `int order_accuracy_`
 - Order of accuracy.*
- `Real mimetic_threshold_`
 - Mimetic Threshold.*

17.12.1 Detailed Description

This class implements a 2D interpolation operator.

Definition at line 76 of file `mtk_interp_2d.h`.

17.12.2 Constructor & Destructor Documentation

17.12.2.1 `mtk::Interp2D::Interp2D ()`

17.12.2.2 `mtk::Interp2D::Interp2D (const Interp2D & interp)`

Parameters

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

17.12.2.3 `mtk::Interp2D::~Interp2D ()`

17.12.3 Member Function Documentation

17.12.3.1 `DenseMatrix mtk::Interp2D::ConstructInterp2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

17.12.3.2 DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix()

Returns

The operator as a dense matrix.

17.12.4 Member Data Documentation

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

Definition at line 112 of file [mtk_interp_2d.h](#).

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

Definition at line 116 of file [mtk_interp_2d.h](#).

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

Definition at line 114 of file [mtk_interp_2d.h](#).

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

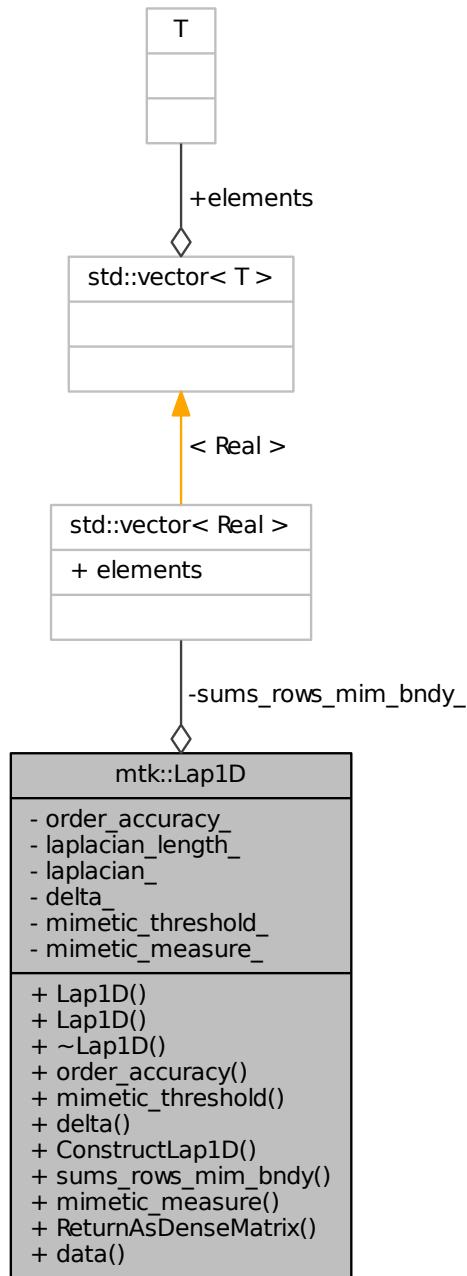
- [include/mtk_interp_2d.h](#)

17.13 mtk::Lap1D Class Reference

Implements a 1D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap1D:



Public Member Functions

- [Lap1D \(\)](#)

- *Default constructor.*
- `Lap1D (const Lap1D &lap)`
Copy constructor.
- `~Lap1D ()`
Destructor.
- `int order_accuracy () const`
Order of accuracy of the operator.
- `Real mimetic_threshold () const`
Mimetic threshold used in the CBS algorithm to construct this operator.
- `Real delta () const`
Value of Δx used be scaled. If 0, then dimensionless.
- `bool ConstructLap1D (int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold)`
Factory method implementing the CBS Algorithm to build operator.
- `std::vector< Real > sums_rows_mim_bndy () const`
Return collection of row-sums mimetic approximations at the boundary.
- `Real mimetic_measure () const`
Returns mimetic measure of the operator.
- `DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid) const`
Return the operator as a dense matrix.
- `const mtk::Real * data (const UniStgGrid1D &grid) const`
Return the operator as a dense array.

Private Attributes

- `int order_accuracy_`
Order of numerical accuracy of the operator.
- `int laplacian_length_`
Length of the output array.
- `Real * laplacian_`
Output array containing the operator and weights.
- `Real delta_`
< If 0.0, then this Laplacian is dimensionless.
- `Real mimetic_threshold_`
< Mimetic threshold.
- `Real mimetic_measure_`
< Mimetic measure.
- `std::vector< Real > sums_rows_mim_bndy_`
Sum of each mimetic boundary row.

Friends

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

17.13.1 Detailed Description

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

Definition at line 78 of file [mtk_lap_1d.h](#).

17.13.2 Constructor & Destructor Documentation

17.13.2.1 mtk::Lap1D::Lap1D()

Definition at line 114 of file [mtk_lap_1d.cc](#).

17.13.2.2 mtk::Lap1D::Lap1D(const Lap1D & lap)

Parameters

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

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

17.13.2.3 mtk::Lap1D::~Lap1D()

Definition at line 130 of file [mtk_lap_1d.cc](#).

17.13.3 Member Function Documentation

17.13.3.1 bool mtk::Lap1D::ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)

Returns

Success of the solution.

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

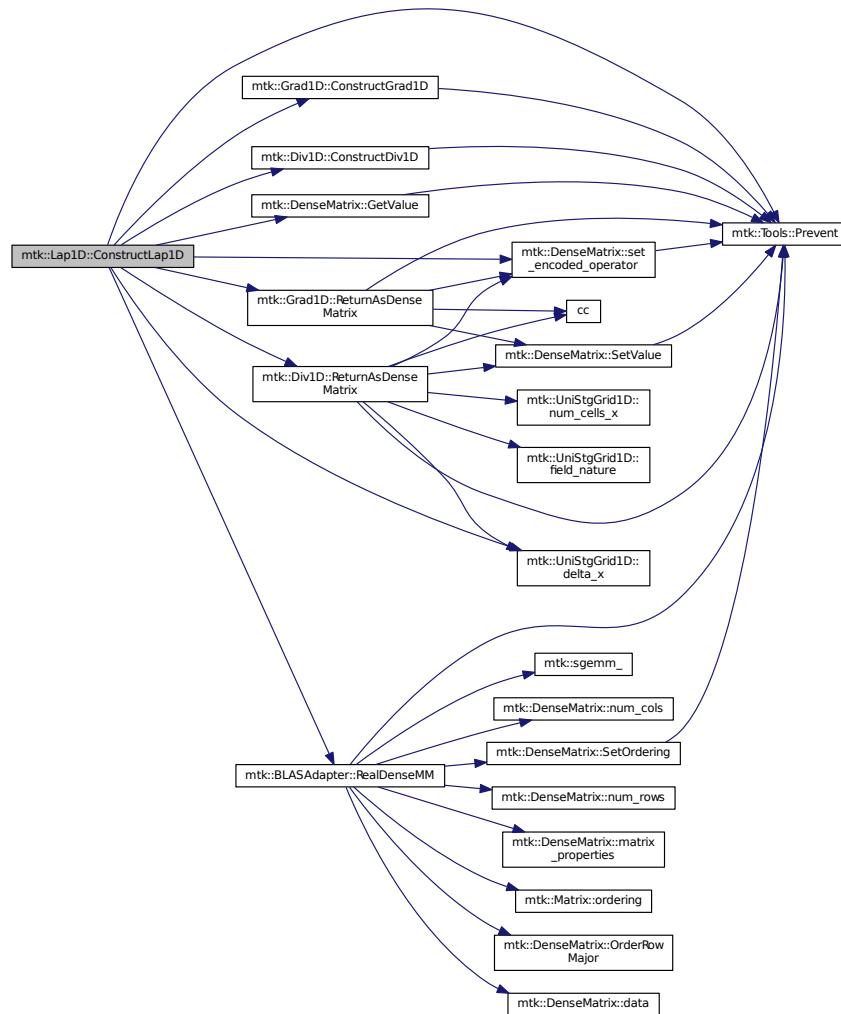
Warning

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

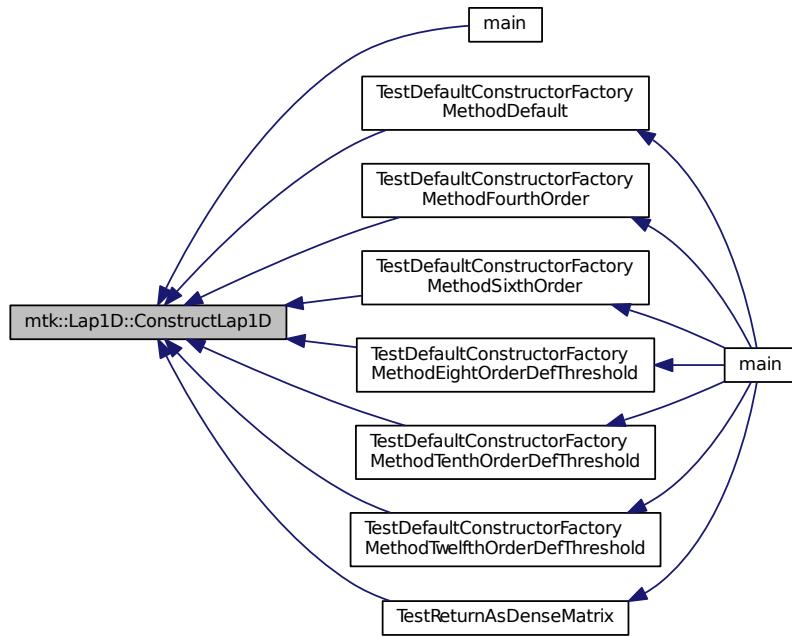
1. The first entry of the array will contain the order of accuracy.
2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.
3. We DO NOT have weights in this operator. Copy and sum mim. bndy coeffs.

Definition at line 151 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



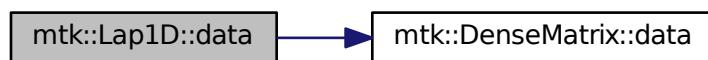
17.13.3.2 const mtk::Real * mtk::Lap1D::data (const UniStgGrid1D & grid) const

Returns

The operator as a dense array.

Definition at line 396 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



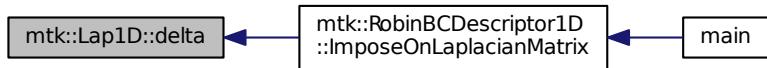
17.13.3.3 mtk::Real mtk::Lap1D::delta () const

Returns

Value of Δx used be scaled. If 0, then dimensionless.

Definition at line 146 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:

**17.13.3.4 mtk::Real mtk::Lap1D::mimetic_measure() const****Returns**

Real number which is the mimetic measure of the operator.

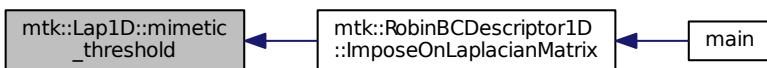
Definition at line 319 of file [mtk_lap_1d.cc](#).

17.13.3.5 mtk::Real mtk::Lap1D::mimetic_threshold() const**Returns**

Mimetic threshold used in the CBS algorithm to construct operator.

Definition at line 141 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:

**17.13.3.6 int mtk::Lap1D::order_accuracy() const**

Returns

Order of accuracy of the operator.

Definition at line 136 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:



17.13.3.7 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix (const UniStgGrid1D & grid) const

Returns

The operator as a dense matrix.

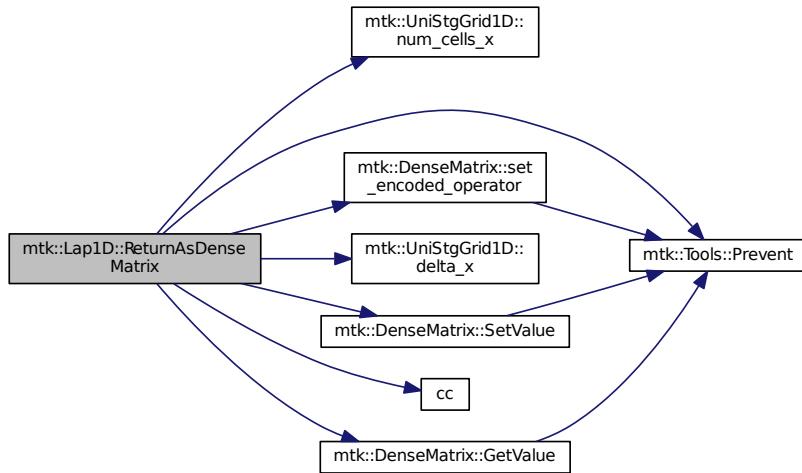
1. Extract mimetic coefficients from the west boundary.
2. Extract interior coefficients.
3. Extract mimetic coefficients from the west boundary to go east.

Note

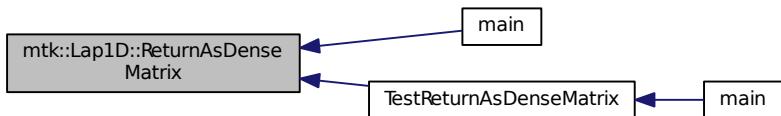
We could create two matrices of the requested size and multiply them, but that would be inefficient, since we already have the computed coefficients stored. We just have to set them in place, in a matrix of an adequate size, and multiply them times the inverse of the square of the step size, in order for the matrix to actually represent a differential operator.

Definition at line 324 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.13.3.8 `std::vector<mtk::Real> mtk::Lap1D::sums_rows_mim_bndy() const`

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 314 of file [mtk_lap_1d.cc](#).

17.13.4 Friends And Related Function Documentation

17.13.4.1 `std::ostream& operator<< (std::ostream & stream, mtk::Lap1D & in)` [friend]

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

Definition at line 75 of file [mtk_lap_1d.cc](#).

17.13.5 Member Data Documentation

17.13.5.1 `Real mtk::Lap1D::delta_` [mutable], [private]

Definition at line 165 of file [mtk_lap_1d.h](#).

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

Definition at line 163 of file [mtk_lap_1d.h](#).

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

Definition at line 161 of file [mtk_lap_1d.h](#).

17.13.5.4 `Real mtk::Lap1D::mimetic_measure_` [private]

Definition at line 168 of file [mtk_lap_1d.h](#).

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

Definition at line 167 of file [mtk_lap_1d.h](#).

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

Definition at line 160 of file [mtk_lap_1d.h](#).

17.13.5.7 `std::vector<Real> mtk::Lap1D::sums_rows_mim_bndy_` [private]

Definition at line 170 of file [mtk_lap_1d.h](#).

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

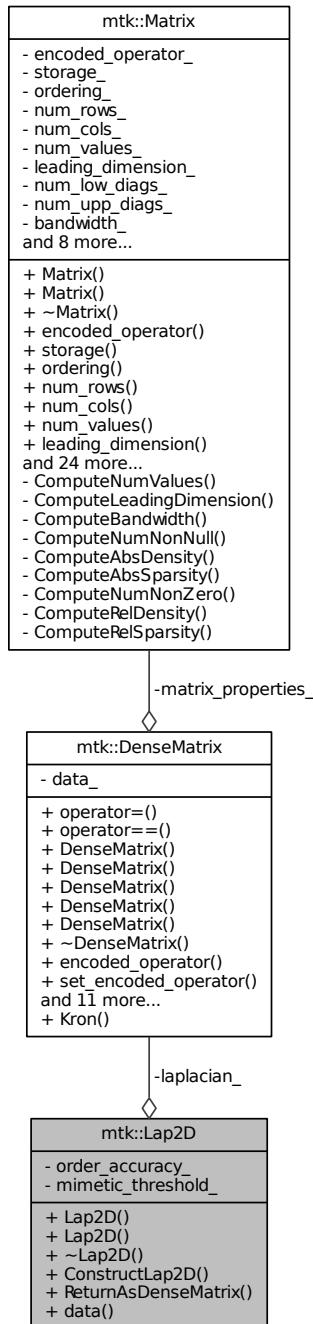
- [include/mtk_lap_1d.h](#)
- [src/mtk_lap_1d.cc](#)

17.14 mtk::Lap2D Class Reference

Implements a 2D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap2D:



Public Member Functions

- [Lap2D \(\)](#)
Default constructor.
- [Lap2D \(const Lap2D &lap\)](#)
Copy constructor.
- [~Lap2D \(\)](#)
Destructor.
- [bool ConstructLap2D \(const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold\)](#)
Factory method implementing the CBS Algorithm to build operator.
- [DenseMatrix ReturnAsDenseMatrix \(\) const](#)
Return the operator as a dense matrix.
- [Real * data \(\) const](#)
Return the operator as a dense array.

Private Attributes

- [DenseMatrix laplacian_](#)
Actual operator.
- [int order_accuracy_](#)
Order of accuracy.
- [Real mimetic_threshold_](#)
Mimetic Threshold.

17.14.1 Detailed Description

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

Definition at line 76 of file [mtk_lap_2d.h](#).

17.14.2 Constructor & Destructor Documentation

17.14.2.1 mtk::Lap2D::Lap2D ()

Definition at line 70 of file [mtk_lap_2d.cc](#).

17.14.2.2 mtk::Lap2D::Lap2D (const Lap2D & lap)

Parameters

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

Definition at line 72 of file [mtk_lap_2d.cc](#).

17.14.2.3 mtk::Lap2D::~Lap2D ()

Definition at line 76 of file [mtk_lap_2d.cc](#).

17.14.3 Member Function Documentation

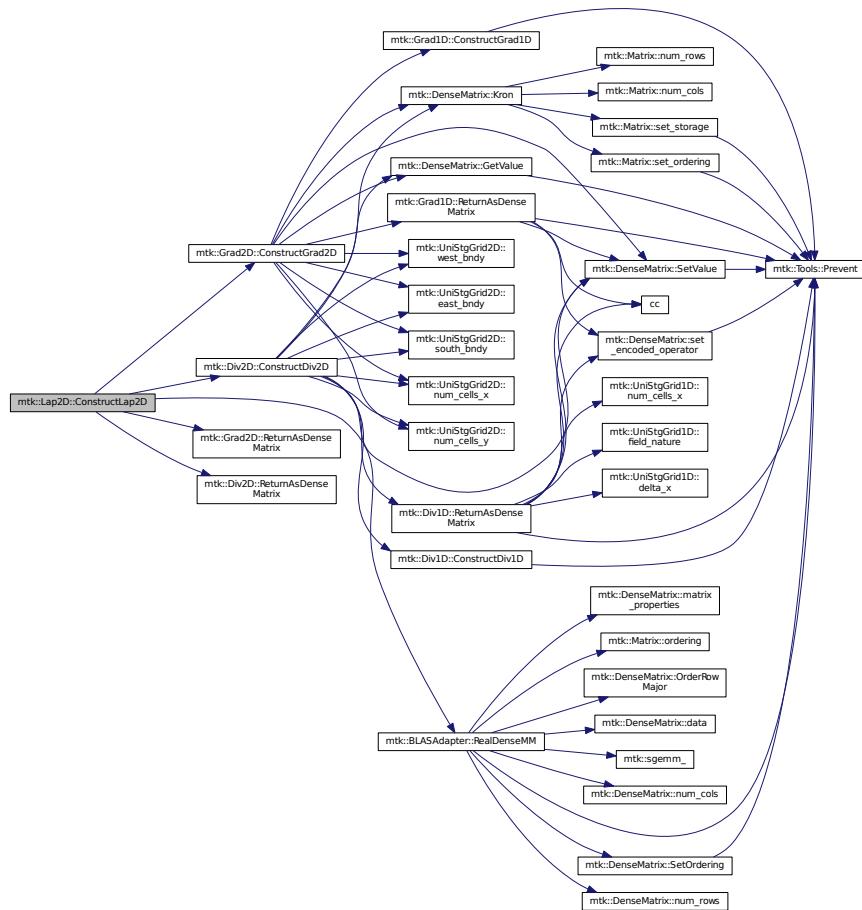
17.14.3.1 `bool mtk::Lap2D::ConstructLap2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

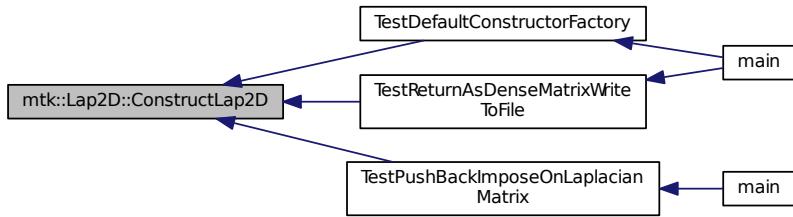
Success of the construction.

Definition at line 78 of file [mtk_lap_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.14.3.2 `mtk::Real * mtk::Lap2D::data() const`

Returns

The operator as a dense array.

Definition at line 116 of file [mtk_lap_2d.cc](#).

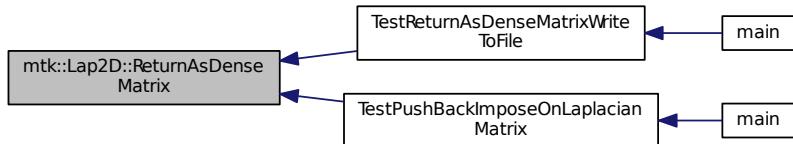
17.14.3.3 `mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix() const`

Returns

The operator as a dense matrix.

Definition at line 111 of file [mtk_lap_2d.cc](#).

Here is the caller graph for this function:



17.14.4 Member Data Documentation

17.14.4.1 `DenseMatrix mtk::Lap2D::laplacian_ [private]`

Definition at line 119 of file [mtk_lap_2d.h](#).

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

Definition at line 123 of file [mtk_lap_2d.h](#).

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

Definition at line 121 of file [mtk_lap_2d.h](#).

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

- include/[mtk_lap_2d.h](#)

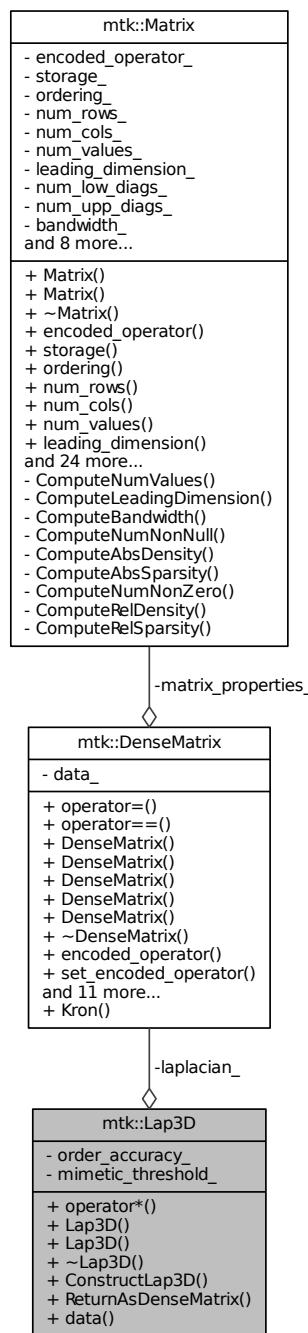
- src/[mtk_lap_2d.cc](#)

17.15 mtk::Lap3D Class Reference

Implements a 3D mimetic Laplacian operator.

```
#include <mtk_lap_3d.h>
```

Collaboration diagram for mtk::Lap3D:



Public Member Functions

- `UniStgGrid3D operator* (const UniStgGrid3D &grid) const`

- Operator application operator on a grid.*
- [Lap3D \(\)](#)
Default constructor.
 - [Lap3D \(const Lap3D &lap\)](#)
Copy constructor.
 - [~Lap3D \(\)](#)
Destructor.
 - [bool ConstructLap3D \(const UniStgGrid3D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimeticThreshold\)](#)
Factory method implementing the CBS Algorithm to build operator.
 - [DenseMatrix ReturnAsDenseMatrix \(\) const](#)
Return the operator as a dense matrix.
 - [Real * data \(\) const](#)
Return the operator as a dense array.

Private Attributes

- [DenseMatrix laplacian_](#)
Actual operator.
- [int order_accuracy_](#)
Order of accuracy.
- [Real mimetic_threshold_](#)
Mimetic Threshold.

17.15.1 Detailed Description

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

Definition at line 76 of file [mtk_lap_3d.h](#).

17.15.2 Constructor & Destructor Documentation

17.15.2.1 [mtk::Lap3D::Lap3D \(\)](#)

Definition at line 76 of file [mtk_lap_3d.cc](#).

17.15.2.2 [mtk::Lap3D::Lap3D \(const Lap3D & lap \)](#)

Parameters

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

Definition at line 78 of file [mtk_lap_3d.cc](#).

17.15.2.3 [mtk::Lap3D::~Lap3D \(\)](#)

Definition at line 82 of file [mtk_lap_3d.cc](#).

17.15.3 Member Function Documentation

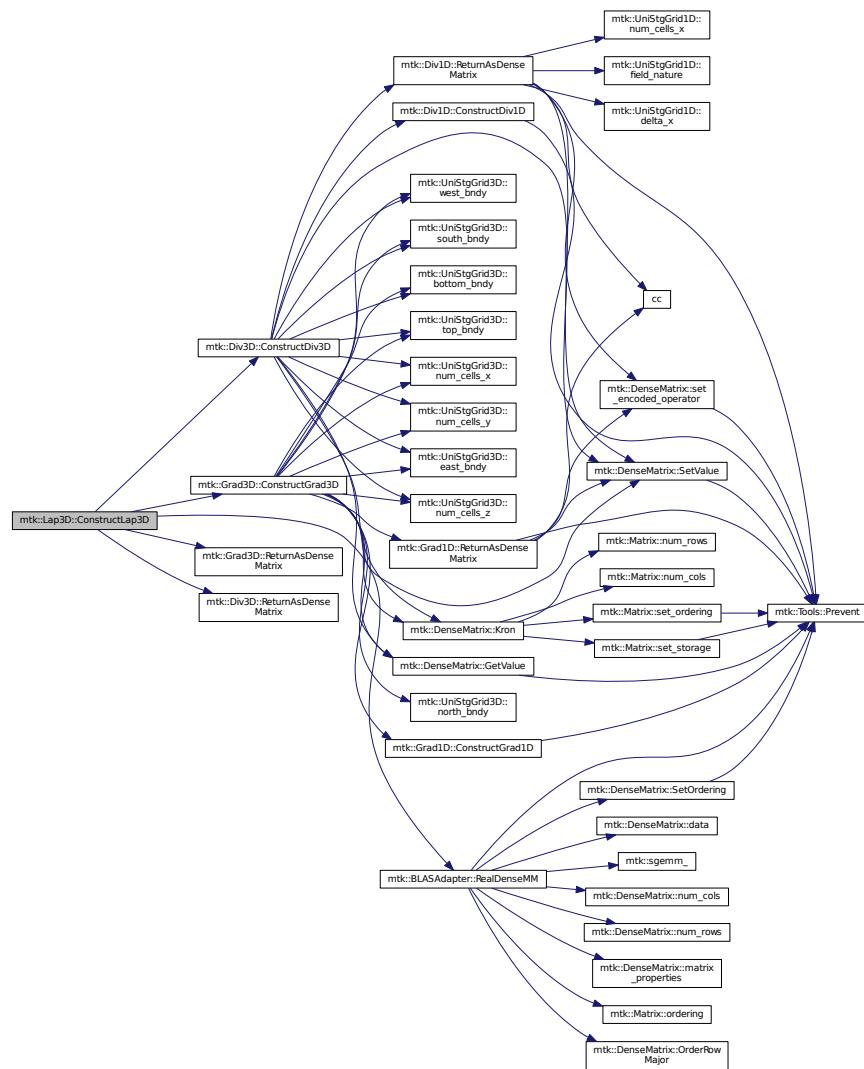
17.15.3.1 `bool mtk::Lap3D::ConstructLap3D (const UniStgGrid3D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

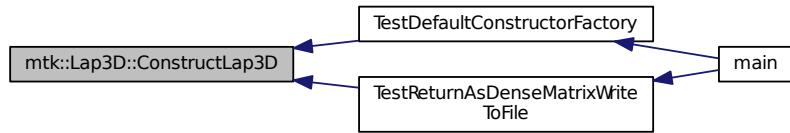
Success of the construction.

Definition at line 84 of file [mtk_lap_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.15.3.2 mtk::Real * mtk::Lap3D::data() const

Returns

The operator as a dense array.

Definition at line 122 of file [mtk_lap_3d.cc](#).

17.15.3.3 mtk::UniStgGrid3D mtk::Lap3D::operator*(const UniStgGrid3D & grid) const

Definition at line 69 of file [mtk_lap_3d.cc](#).

17.15.3.4 mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix() const

Returns

The operator as a dense matrix.

Definition at line 117 of file [mtk_lap_3d.cc](#).

Here is the caller graph for this function:



17.15.4 Member Data Documentation

17.15.4.1 DenseMatrix mtk::Lap3D::laplacian_ [private]

Definition at line 124 of file [mtk_lap_3d.h](#).

17.15.4.2 Real mtk::Lap3D::mimetic_threshold_ [private]

Definition at line 128 of file [mtk_lap_3d.h](#).

17.15.4.3 int mtk::Lap3D::order_accuracy_ [private]

Definition at line 126 of file [mtk_lap_3d.h](#).

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

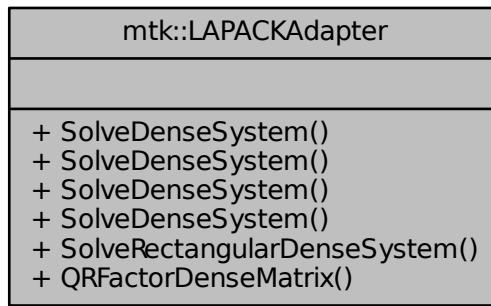
- [include/mtk_lap_3d.h](#)
- [src/mtk_lap_3d.cc](#)

17.16 mtk::LAPACKAdapter Class Reference

Adapter class for the LAPACK API.

```
#include <mtk_lapack_adapter.h>
```

Collaboration diagram for mtk::LAPACKAdapter:



Static Public Member Functions

- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::Real](#) *rhs)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::DenseMatrix](#) &rr)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid1D](#) &rhs)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid2D](#) &rhs)
Solves a dense system of linear equations.
- static int [SolveRectangularDenseSystem](#) (const [mtk::DenseMatrix](#) &aa, [mtk::Real](#) *ob_, int ob_Id_)
Solves overdetermined or underdetermined real linear systems.
- static [mtk::DenseMatrix](#) [QRFactorDenseMatrix](#) ([DenseMatrix](#) &matrix)
Performs a QR factorization on a dense matrix.

17.16.1 Detailed Description

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

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Definition at line 94 of file [mtk_lapack_adapter.h](#).

17.16.2 Member Function Documentation

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

Adapts the MTK to LAPACK's routine.

Parameters

in, out	matrix	Input matrix.
---------	--------	---------------

Returns

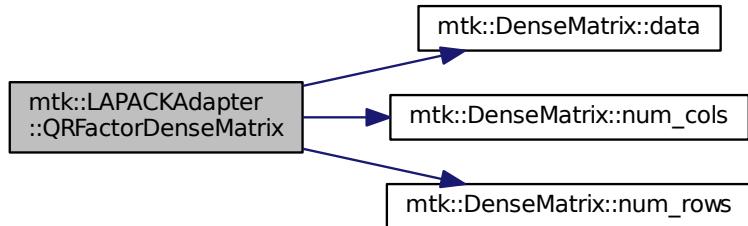
Matrix **Q**.

Exceptions

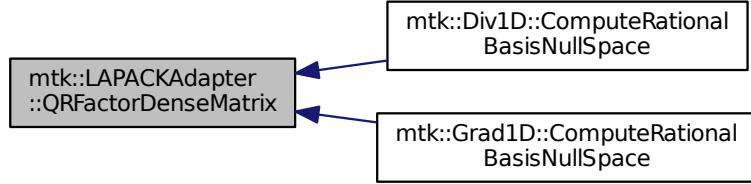
<code>std::bad_alloc</code>

Definition at line 595 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

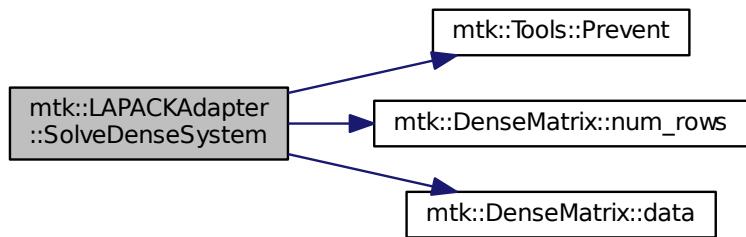
in	<i>matrix</i>	Input matrix.
in	<i>rhs</i>	Input right-hand sides vector.

Exceptions

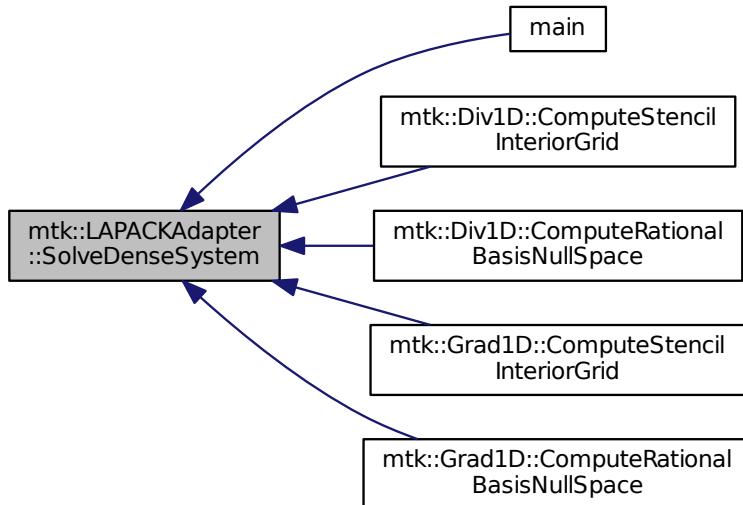
<i>std::bad_alloc</i>

Definition at line 432 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

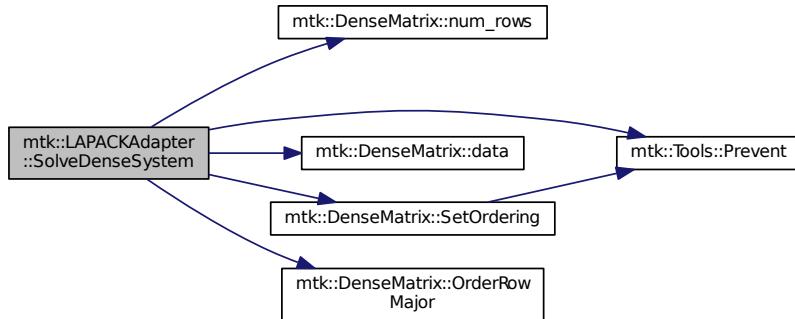
in	<i>matrix</i>	Input matrix.
in	<i>rr</i>	Input right-hand sides matrix.

Exceptions

<i>std::bad_alloc</i>

Definition at line 467 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



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

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

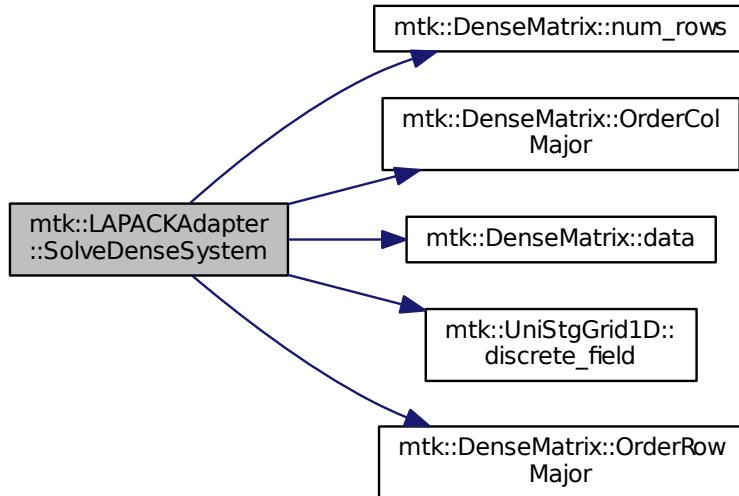
in	<i>matrix</i>	Input matrix.
in	<i>rhs</i>	Input right-hand side from info on a grid.

Exceptions

<code>std::bad_alloc</code>

Definition at line 519 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



17.16.2.5 `int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & mm, mtk::UniStgGrid2D & rhs) [static]`

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

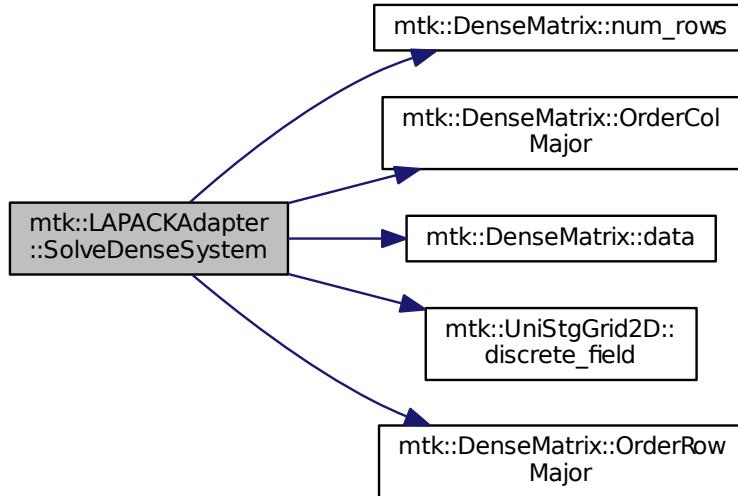
in	<i>matrix</i>	Input matrix.
in	<i>rhs</i>	Input right-hand side from info on a grid.

Exceptions

<code>std::bad_alloc</code>

Definition at line 557 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



17.16.2.6 `int mtk::LAPACKAdapter::SolveRectangularDenseSystem (const mtk::DenseMatrix & aa, mtk::Real * ob_, int ob_id_) [static]`

Adapts the MTK to LAPACK's routine.

Parameters

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

Returns

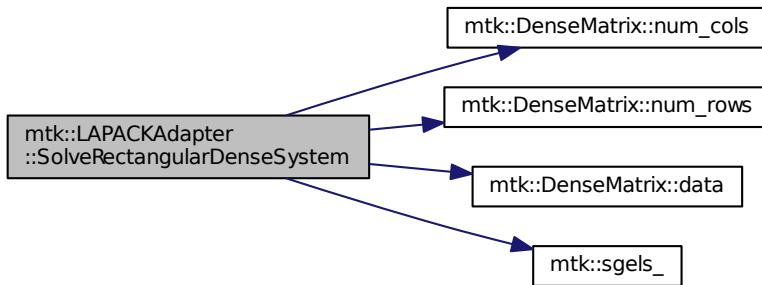
Success of the solution.

Exceptions

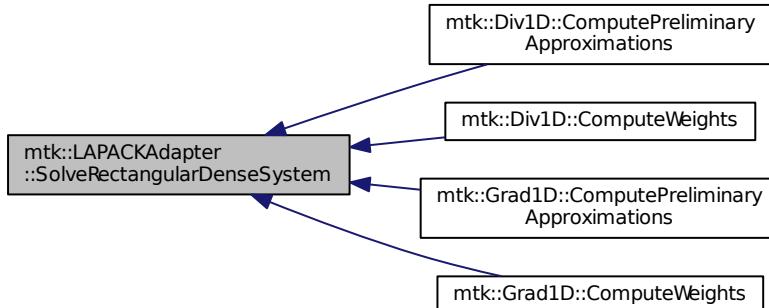
<code>std::bad_alloc</code>

Definition at line 792 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

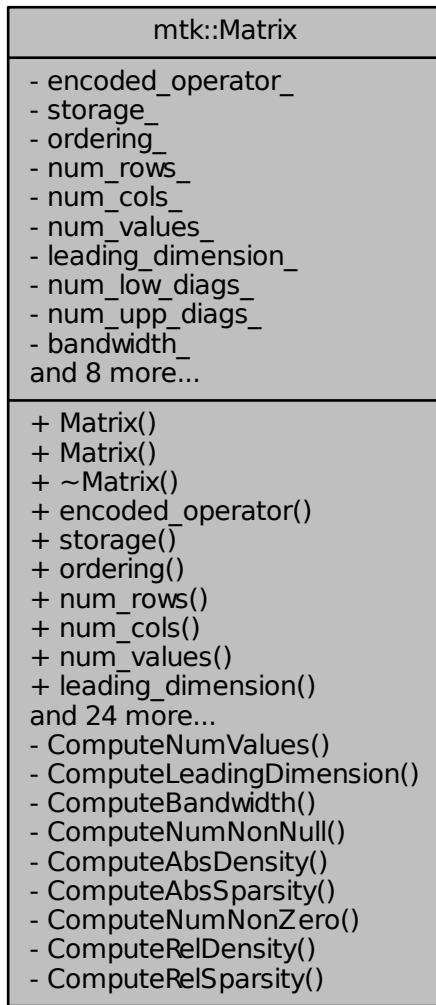
- include/[mtk_lapack_adapter.h](#)
- src/[mtk_lapack_adapter.cc](#)

17.17 mtk::Matrix Class Reference

Definition of the representation of a matrix in the MTK.

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

Collaboration diagram for mtk::Matrix:



Public Member Functions

- [Matrix \(\)](#)
Default constructor.
- [Matrix \(const Matrix &in\)](#)
Copy constructor.
- [~Matrix \(\) noexcept](#)
Destructor.
- [EncodedOperator encoded_operator \(\) const noexcept](#)

- Gets the type of mimetic operator encoded by this matrix.
- **MatrixStorage storage () const noexcept**
Gets the type of storage of this matrix.
- **MatrixOrdering ordering () const noexcept**
Gets the type of ordering of this matrix.
- int **num_rows () const noexcept**
Gets the number of rows.
- int **num_cols () const noexcept**
Gets the number of columns.
- int **num_values () const noexcept**
Gets the number of values.
- int **leading_dimension () const noexcept**
Gets the matrix' leading dimension.
- int **num_zero () const noexcept**
Gets the number of zeros.
- int **num_non_zero () const noexcept**
Gets the number of non-zero values.
- int **num_null () const noexcept**
Gets the number of null values.
- int **num_non_null () const noexcept**
Gets the number of non-null values.
- int **num_low_diags () const noexcept**
Gets the number of lower diagonals.
- int **num_upp_diags () const noexcept**
Gets the number of upper diagonals.
- int **bandwidth () const noexcept**
Gets the bandwidth.
- **Real abs_density () const noexcept**
Gets the absolute density.
- **Real rel_density () const noexcept**
Gets the relative density.
- **Real abs_sparsity () const noexcept**
Gets the Absolute sparsity.
- **Real rel_sparsity () const noexcept**
Gets the Relative sparsity.
- void **set_encoded_operator (const EncodedOperator &in) noexcept**
Sets the type of encoded operator of the matrix.
- void **set_storage (const MatrixStorage &tt) noexcept**
Sets the storage type of the matrix.
- void **set_ordering (const MatrixOrdering &oo) noexcept**
Sets the ordering of the matrix.
- void **set_num_rows (const int &num_rows) noexcept**
Sets the number of rows of the matrix.
- void **set_num_cols (const int &num_cols) noexcept**
Sets the number of columns of the matrix.
- void **set_num_low_diags (const int &num_low_diags) noexcept**
Sets the number of lower diagonals of the matrix.

- void `set_num_upp_diags` (const int &`num_upp_diags`) noexcept
Sets the number of upper diagonals of the matrix.
- void `set_num_null` (const int &`num_null`) noexcept
Sets the number of null elements of the matrix.
- void `set_num_zero` (const int &`num_zero`) noexcept
Sets the number of zeros of the matrix.
- void `IncreaseNumNull` () noexcept
Decreases the number of values that equal zero but with no meaning.
- void `DecreaseNumNull` () noexcept
Decreases the number of values that equal zero but with no meaning.
- void `IncreaseNumZero` () noexcept
Increases the number of values that equal zero but with meaning.
- void `DecreaseNumZero` () noexcept
Decreases the number of values that equal zero but with meaning.

Private Member Functions

- int `ComputeNumValues` (const int &`num_rows`, const int &`num_cols`) const noexcept
Computes the leading dimension of the matrix.
- int `ComputeLeadingDimension` (const int &`num_rows`, const int &`num_cols`) const noexcept
Computes the leading dimension of the matrix.
- int `ComputeBandwidth` (const int &`num_low_diags`, const int &`num_upp_diags`) const noexcept
Computes the bandwidth of the matrix.
- int `ComputeNumNonNull` (const int &`num_values`, const int &`num_null`) const noexcept
Computes the number of non-null values of the matrix.
- `mtk::Real ComputeAbsDensity` (const int &`num_non_null`, const int &`num_values`) const noexcept
Computes the absolute density of the matrix.
- `mtk::Real ComputeAbsSparsity` (const `mtk::Real` &`absolute_density`) const noexcept
Computes the absolute sparsity of the matrix.
- int `ComputeNumNonZero` (const int &`num_values`, const int &`num_zero`) const noexcept
Computes the number of non-zero values of the matrix.
- `mtk::Real ComputeRelDensity` (const int &`num_non_zero`, const int &`num_values`) const noexcept
Computes the relative density of the matrix.
- `mtk::Real ComputeRelSparsity` (const `mtk::Real` &`relative_density`) const noexcept
Computes the relative sparsity of the matrix.

Private Attributes

- `EncodedOperator encoded_operator_`
- `MatrixStorage storage_`
Type of mimetic operator encoded.
- `MatrixOrdering ordering_`
What kind of ordering is it following?
- int `num_rows_`
Number of rows.
- int `num_cols_`

- int `num_values_`
Number of columns.
- int `leading_dimension_`
Number of total values in matrix.
- int `num_low_diags_`
Elements between successive rows when row-major.
- int `num_upp_diags_`
Number of lower diagonals on a banded matrix.
- int `num_upp_diags_`
Number of upper diagonals on a banded matrix.
- int `bandwidth_`
Bandwidth of the matrix.
- int `num_null_`
Number of null (no meaning) values.
- int `num_non_null_`
Number of null values.
- Real `abs_density_`
Absolute density of matrix.
- Real `abs_sparsity_`
Absolute sparsity of matrix.
- int `num_zero_`
Number of zeros (with meaning).
- int `num_non_zero_`
Number of non-zero values.
- Real `rel_density_`
Relative density of matrix.
- Real `rel_sparsity_`
Relative sparsity of matrix.

17.17.1 Detailed Description

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

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

17.17.2 Constructor & Destructor Documentation

17.17.2.1 `mtk::Matrix::Matrix()`

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

17.17.2.2 `mtk::Matrix::Matrix(const Matrix & in)`

Parameters

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

Definition at line 87 of file [mtk_matrix.cc](#).

17.17.2.3 mtk::Matrix::~Matrix() [noexcept]

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

17.17.3 Member Function Documentation

17.17.3.1 mtk::Real mtk::Matrix::abs_density() const [noexcept]

Returns

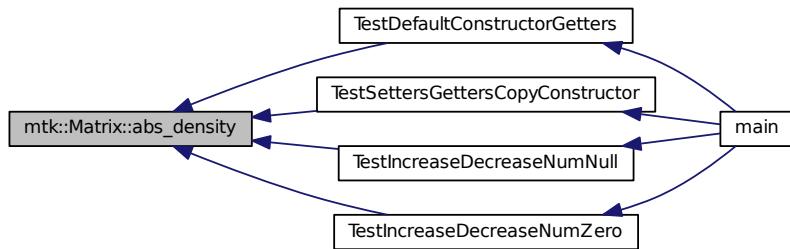
Absolute density of the matrix.

See also

http://www.cs.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 179 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.2 mtk::Real mtk::Matrix::abs_sparsity() const [noexcept]

Returns

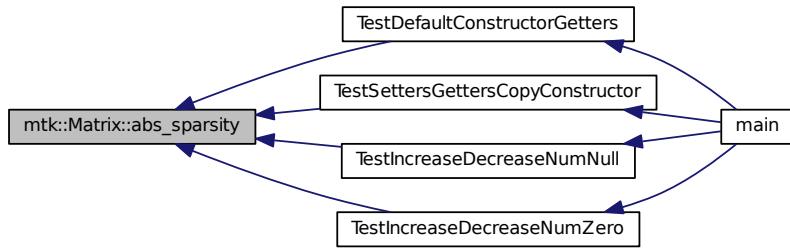
Absolute sparsity of the matrix.

See also

http://www.cs.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 189 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



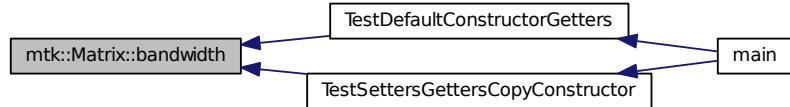
17.17.3.3 int mtk::Matrix::bandwidth () const [noexcept]

Returns

Bandwidth of the matrix.

Definition at line 174 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.4 mtk::Real mtk::Matrix::ComputeAbsDensity (const int & num_non_null, const int & num_values) const [private], [noexcept]

Defined as

$$\frac{\text{num_non_null}}{\text{num_values}}.$$

Parameters

in	<i>num_non_null</i>	Number of non-null values of the matrix.
in	<i>num_values</i>	Number of total values of the matrix.

Returns

Absolute density of the matrix.

Definition at line 434 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.5 mtk::Real mtk::Matrix::ComputeAbsSparsity (const mtk::Real & *absolute_density*) const [private], [noexcept]

Defined as

$$1 - \frac{\text{num_non_null}}{\text{num_values}}.$$

Parameters

in	<i>absolute_density</i>	Absolute density of the matrix.
----	-------------------------	---------------------------------

Returns

Absolute sparsity of the matrix.

Definition at line 446 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.6 int mtk::Matrix::ComputeBandwidth (const int & *num_low_diags*, const int & *num_upp_diags*) const [private], [noexcept]

Parameters

in	<i>num_low_diags</i>	Number of lower diagonals.
in	<i>num_upp_diags</i>	Number of upper diagonals.

Returns

Bandwidth of the matrix.

Definition at line 412 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.7 int mtk::Matrix::ComputeLeadingDimension (const int & *num_rows*, const int & *num_cols*) const [private], [noexcept]

Parameters

in	<i>num_rows</i>	Number of rows.
in	<i>num_cols</i>	Number of columns.

Returns

Leading dimension of the matrix.

Definition at line 400 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.8 int mtk::Matrix::ComputeNumNonNull (const int & *num_values*, const int & *num_null*) const [private], [noexcept]

Parameters

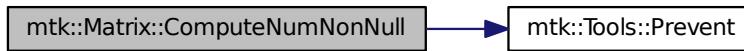
in	<i>num_values</i>	Number of values of the matrix.
in	<i>num_null</i>	Number of null values of the matrix.

Returns

Number of non-null values of the matrix.

Definition at line 423 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.9 `int mtk::Matrix::ComputeNumNonZero (const int & num_values, const int & num_zero) const [private], [noexcept]`

Parameters

in	<i>num_values</i>	Number of values of the matrix.
in	<i>num_zero</i>	Number of zero values of the matrix.

Returns

Number of non-zero values of the matrix.

Definition at line 456 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.10 `int mtk::Matrix::ComputeNumValues (const int & num_rows, const int & num_cols) const [private], [noexcept]`

Parameters

in	<i>num_rows</i>	Number of rows.
in	<i>num_cols</i>	Number of columns.

Returns

Number of values of the matrix.

Definition at line 389 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.11 **mtk::Real** **mtk::Matrix::ComputeRelDensity** (**const int & num_non_zero, const int & num_values**) **const**
[private], [noexcept]

Defined as

$$\frac{\text{num_non_zero}}{\text{num_values}}.$$

Parameters

in	<i>num_non_zero</i>	Number of non-zero values of the matrix.
in	<i>num_values</i>	Number of total values of the matrix.

Returns

Relative density of the matrix.

Definition at line 467 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.12 `mtk::Real mtk::Matrix::ComputeRelSparsity (const mtk::Real & relative_density) const` [private],
 [noexcept]

Defined as

$$1 - \frac{\text{num_non_zero}}{\text{num_values}}.$$

Parameters

<code>in</code>	<code>relative_density</code>	Relative density of the matrix.
-----------------	-------------------------------	---------------------------------

Returns

Relative sparsity of the matrix.

Definition at line 479 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



17.17.3.13 `void mtk::Matrix::DecreaseNumNull()` [noexcept]

Definition at line 346 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



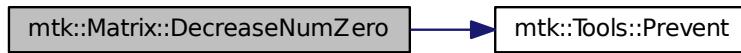
Here is the caller graph for this function:



17.17.3.14 void mtk::Matrix::DecreaseNumZero() [noexcept]

Definition at line 374 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**17.17.3.15 mtk::EncodedOperator mtk::Matrix::encoded_operator() const [noexcept]****Returns**

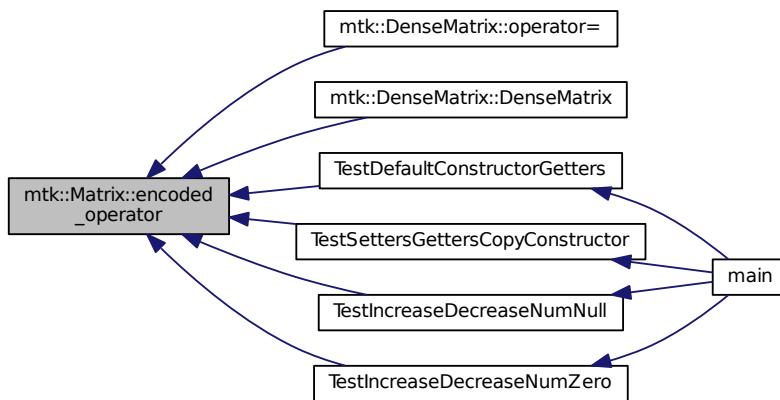
Type of mimetic operator encoded by this matrix.

See also

[mtk::EncodedOperator](#).

Definition at line 109 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.16 void mtk::Matrix::IncreaseNumNull() [noexcept]

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

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.17 void mtk::Matrix::IncreaseNumZero() [noexcept]

Definition at line 359 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.18 int mtk::Matrix::leading_dimension() const [noexcept]

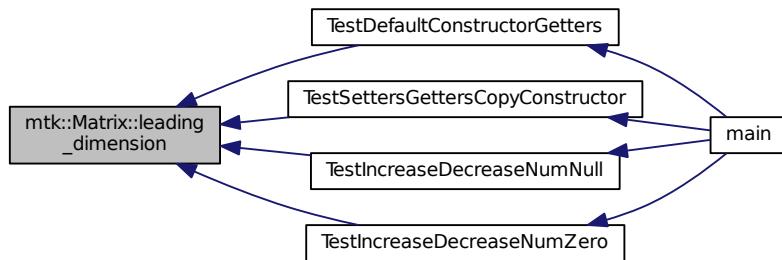
Leading dimension of the data array is the number of elements between successive rows (for row major storage) in memory. Most of the cases, the leading dimension is the same as the number of columns.

Returns

Leading dimension of the matrix.

Definition at line 139 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



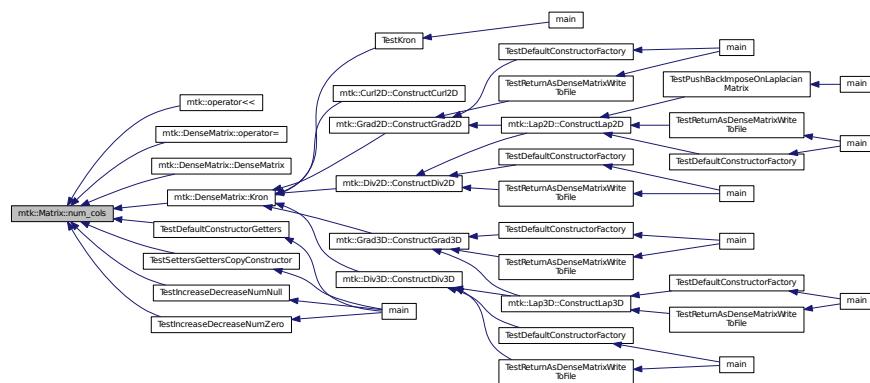
17.17.3.19 int mtk::Matrix::num_cols() const [noexcept]

Returns

Number of rows of the matrix.

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

Here is the caller graph for this function:



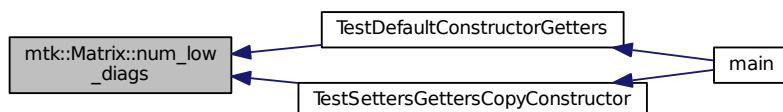
17.17.3.20 int mtk::Matrix::num_low_diags() const [noexcept]

Returns

Number of lower diagonals.

Definition at line 164 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.21 int mtk::Matrix::num_non_null() const [noexcept]

Returns

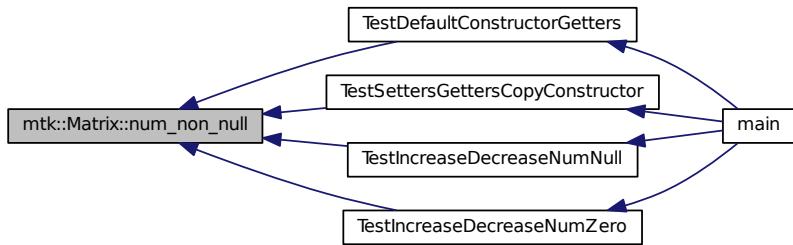
Number of non-null values of the matrix.

See also

http://www.cs.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 159 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



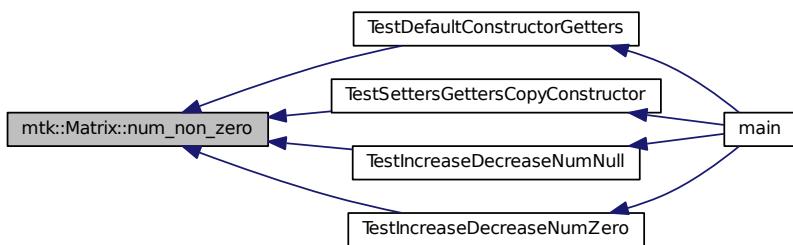
17.17.3.22 `int mtk::Matrix::num_non_zero() const [noexcept]`

Returns

Number of non-zero values of the matrix.

Definition at line 149 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.23 `int mtk::Matrix::num_null() const [noexcept]`

Returns

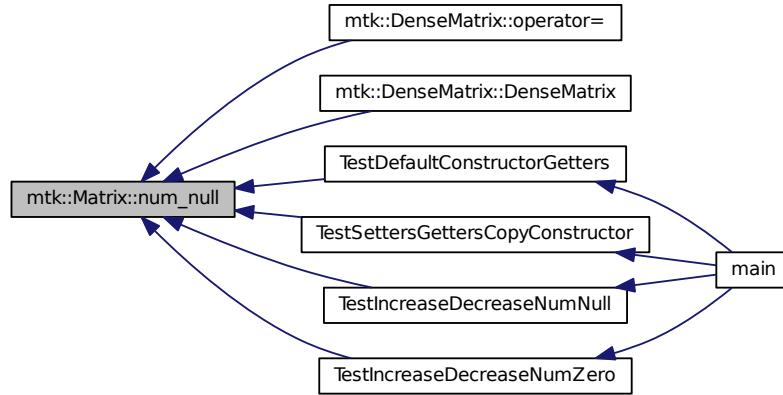
Number of null values of the matrix.

See also

http://www.csdc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 154 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



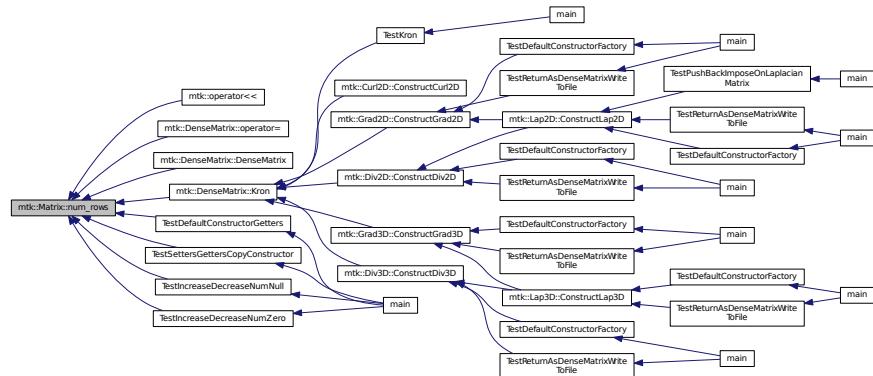
17.17.3.24 int mtk::Matrix::num_rows() const [noexcept]

Returns

Number of rows of the matrix.

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

Here is the caller graph for this function:



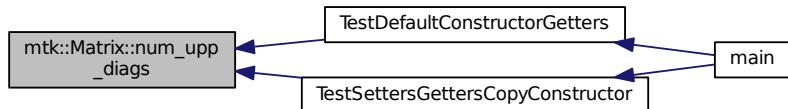
17.17.3.25 int mtk::Matrix::num_upp_diags() const [noexcept]

Returns

Number of upper diagonals.

Definition at line 169 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



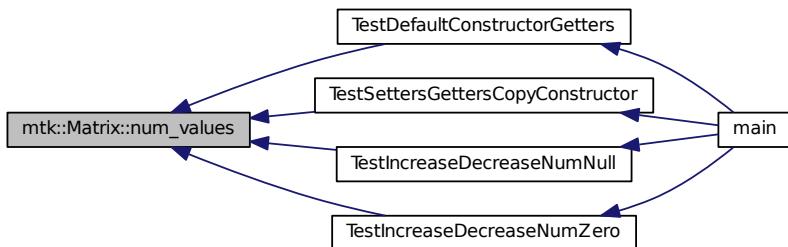
17.17.3.26 int mtk::Matrix::num_values() const [noexcept]

Returns

Number of values of the matrix.

Definition at line 134 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



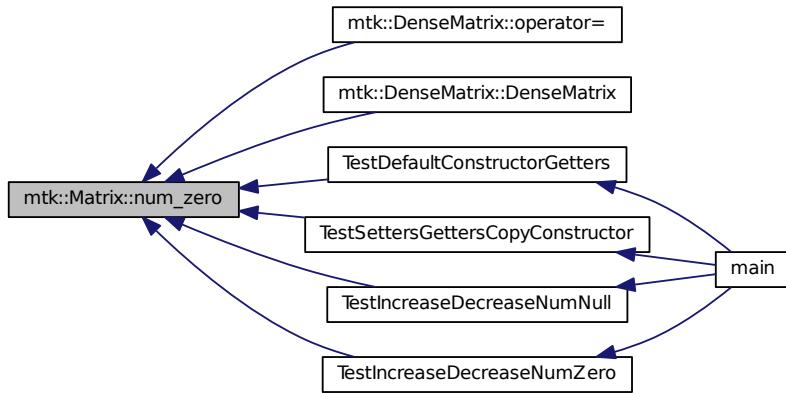
17.17.3.27 int mtk::Matrix::num_zero() const [noexcept]

Returns

Number of zeros of the matrix.

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

Here is the caller graph for this function:

**17.17.3.28 mtk::MatrixOrdering mtk::Matrix::ordering() const [noexcept]****Returns**

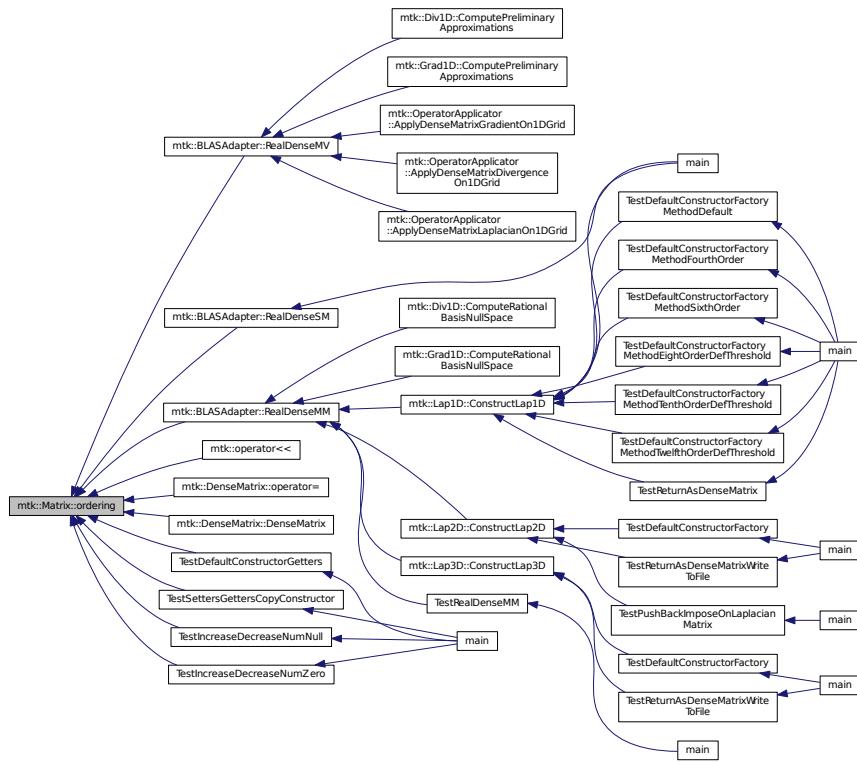
Type of ordering of this matrix.

See also

[mtk::MatrixOrdering](#).

Definition at line 119 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.29 `mtk::Real mtk::Matrix::rel_density() const [noexcept]`

Returns

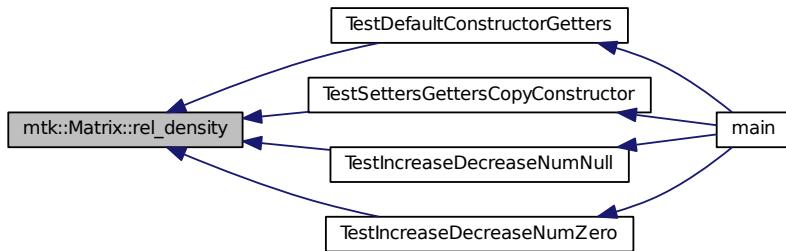
Relative density of the matrix.

See also

http://www.cs.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 184 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.30 mtk::Real mtk::Matrix::rel_sparsity () const [noexcept]

Returns

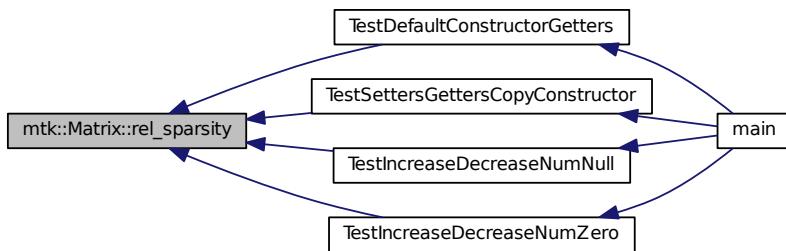
Relative sparsity of the matrix.

See also

http://www.cs.sdsu.edu/research_reports/CSRCR2013-01.pdf

Definition at line 194 of file [mtk_matrix.cc](#).

Here is the caller graph for this function:



17.17.3.31 void mtk::Matrix::set_encoded_operator (const EncodedOperator & in) [noexcept]

Parameters

in	<i>in</i>	Type of encoded operator.
----	-----------	---------------------------

See also

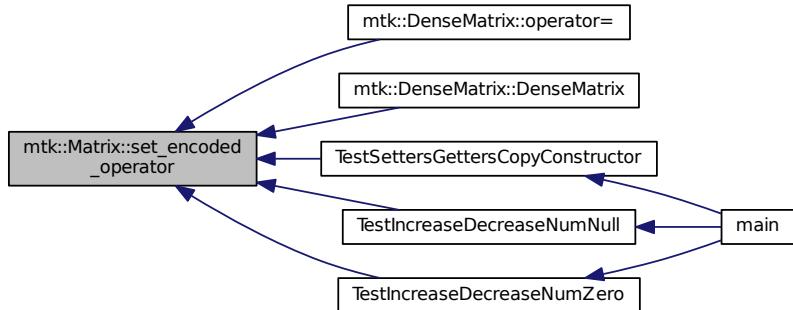
[mtk::EncodedOperator](#)

Definition at line 199 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.32 void mtk::Matrix::set_num_cols(const int & num_cols) [noexcept]

Parameters

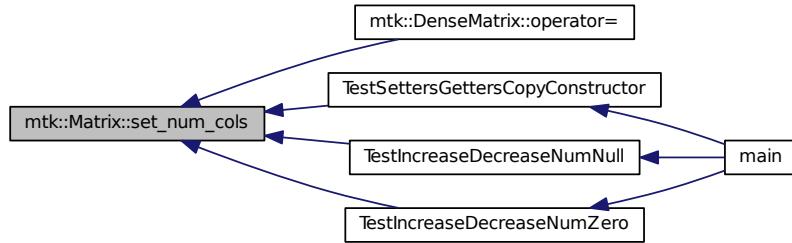
in	<i>num_cols</i>	Number of columns.
----	-----------------	--------------------

Definition at line 262 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



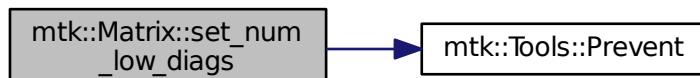
17.17.3.33 void mtk::Matrix::set_num_low_diags (const int & num_low_diags) [noexcept]

Parameters

in	<i>num_low_diags</i>	Number of lower diagonals.
----	----------------------	----------------------------

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

Here is the call graph for this function:



Here is the caller graph for this function:



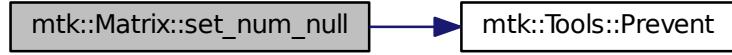
17.17.3.34 void mtk::Matrix::set_num_null (const int & num_null) [noexcept]

Parameters

in	num_null	Number of null elements.
----	----------	--------------------------

Definition at line 303 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.35 void mtk::Matrix::set_num_rows (const int & num_rows) [noexcept]

Parameters

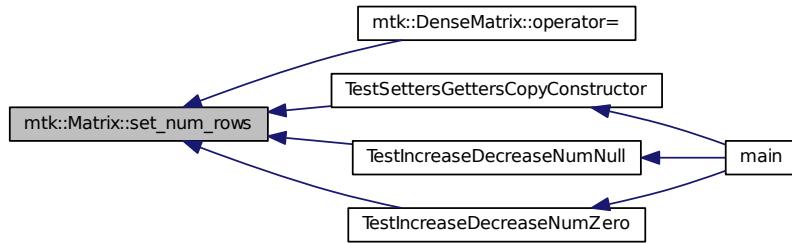
in	num_rows	Number of rows.
----	----------	-----------------

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:



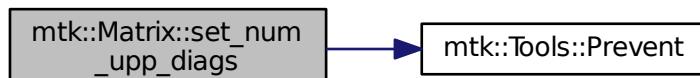
17.17.3.36 void mtk::Matrix::set_num_upp_diags (const int & num_upp_diags) [noexcept]

Parameters

in	<i>num_upp_diags</i>	Number of upper diagonals.
----	----------------------	----------------------------

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

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.37 void mtk::Matrix::set_num_zero (const int & num_zero) [noexcept]

Parameters

in	num_zero	Number of zeros on the matrix.
----	----------	--------------------------------

Definition at line 316 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.38 void mtk::Matrix::set_ordering (const MatrixOrdering & oo) [noexcept]

Parameters

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

See also

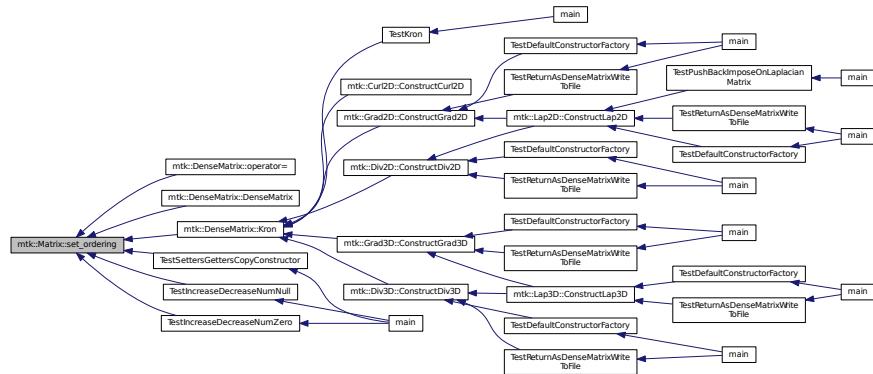
[mtk::MatrixOrdering](#)

Definition at line 228 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.39 void mtk::Matrix::set_storage (const MatrixStorage & tt) [noexcept]

Parameters

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

See also

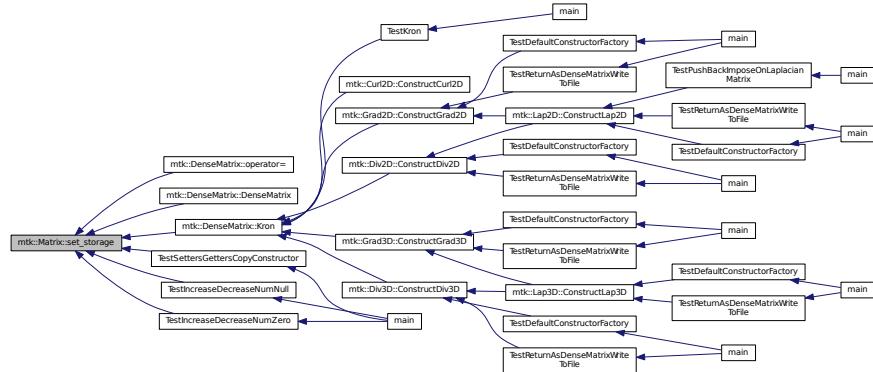
mtk::MatrixStorage

Definition at line 216 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.40 mtk::MatrixStorage mtk::Matrix::storage() const [noexcept]

Returns

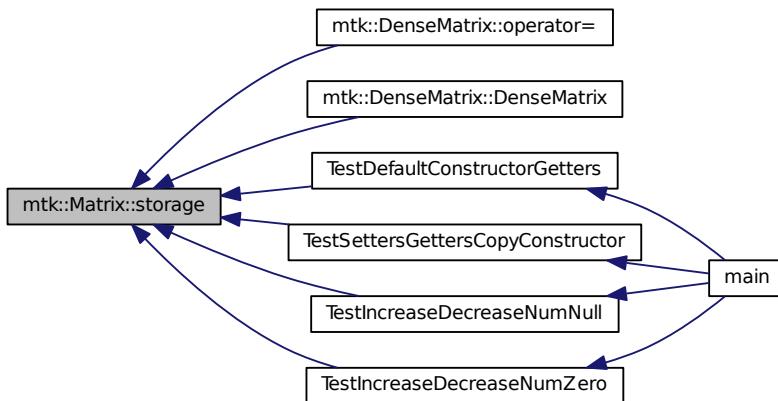
Type of storage of this matrix.

See also

[mtk::MatrixStorage](#).

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

Here is the caller graph for this function:



17.17.4 Member Data Documentation

17.17.4.1 Real mtk::Matrix::abs_density_ [private]

Definition at line 466 of file [mtk_matrix.h](#).

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

Definition at line 467 of file [mtk_matrix.h](#).

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

Definition at line 461 of file [mtk_matrix.h](#).

17.17.4.4 EncodedOperator mtk::Matrix::encoded_operator_ [private]

Definition at line 448 of file [mtk_matrix.h](#).

17.17.4.5 int mtk::Matrix::leading_dimension_ [private]

Definition at line 457 of file [mtk_matrix.h](#).

17.17.4.6 `int mtk::Matrix::num_cols_ [private]`

Definition at line 455 of file [mtk_matrix.h](#).

17.17.4.7 `int mtk::Matrix::num_low_diags_ [private]`

Definition at line 459 of file [mtk_matrix.h](#).

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

Definition at line 464 of file [mtk_matrix.h](#).

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

Definition at line 470 of file [mtk_matrix.h](#).

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

Definition at line 463 of file [mtk_matrix.h](#).

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

Definition at line 454 of file [mtk_matrix.h](#).

17.17.4.12 `int mtk::Matrix::num_upp_diags_ [private]`

Definition at line 460 of file [mtk_matrix.h](#).

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

Definition at line 456 of file [mtk_matrix.h](#).

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

Definition at line 469 of file [mtk_matrix.h](#).

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

Definition at line 452 of file [mtk_matrix.h](#).

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

Definition at line 472 of file [mtk_matrix.h](#).

17.17.4.17 Real mtk::Matrix::rel_sparsity_ [private]

Definition at line 473 of file [mtk_matrix.h](#).

17.17.4.18 MatrixStorage mtk::Matrix::storage_ [private]

What type of matrix is this?

Definition at line 450 of file [mtk_matrix.h](#).

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

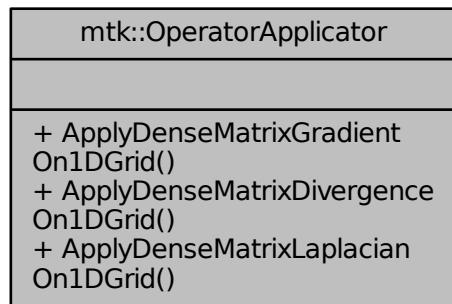
- [include/mtk_matrix.h](#)
- [src/mtk_matrix.cc](#)

17.18 mtk::OperatorApplicator Class Reference

Controls the process of applying a mimetic operator to a grid.

```
#include <mtk_operator_applicator.h>
```

Collaboration diagram for mtk::OperatorApplicator:



Static Public Member Functions

- static void [ApplyDenseMatrixGradientOn1DGrid](#) ([DenseMatrix](#) &grad, [UniStgGrid1D](#) &grid, [UniStgGrid1D](#) &out)
Applies a gradient operator encoded as a dense matrix to a 1D grid.
- static void [ApplyDenseMatrixDivergenceOn1DGrid](#) ([DenseMatrix](#) &div, [UniStgGrid1D](#) &grid, [UniStgGrid1D](#) &out)
Applies a divergence operator encoded as a dense matrix to a 1D grid.
- static void [ApplyDenseMatrixLaplacianOn1DGrid](#) ([DenseMatrix](#) &lap, [UniStgGrid1D](#) &grid, [UniStgGrid1D](#) &out)
Applies a Laplacian operator encoded as a dense matrix to a 1D grid.

17.18.1 Detailed Description

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

17.18.2 Member Function Documentation

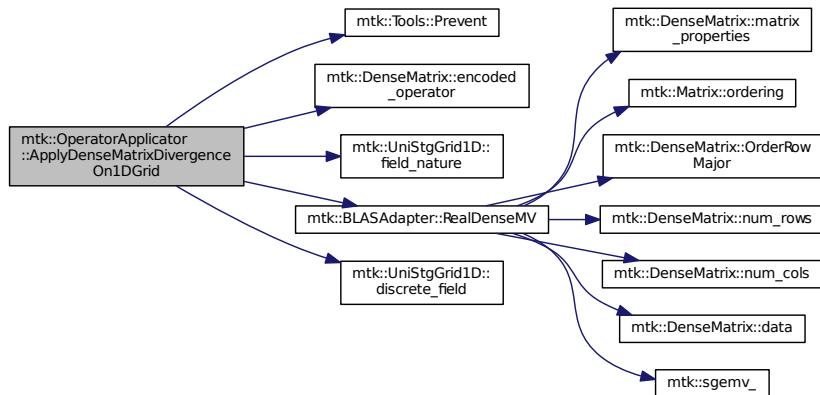
17.18.2.1 void mtk::OperatorApplicator::ApplyDenseMatrixDivergenceOn1DGrid (DenseMatrix & *div*, UniStgGrid1D & *grid*, UniStgGrid1D & *out*) [static]

Parameters

in	<i>div</i>	Divergence operator.
in	<i>grid</i>	1D grid.
out	<i>out</i>	Resulting 1D grid.

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

Here is the call graph for this function:



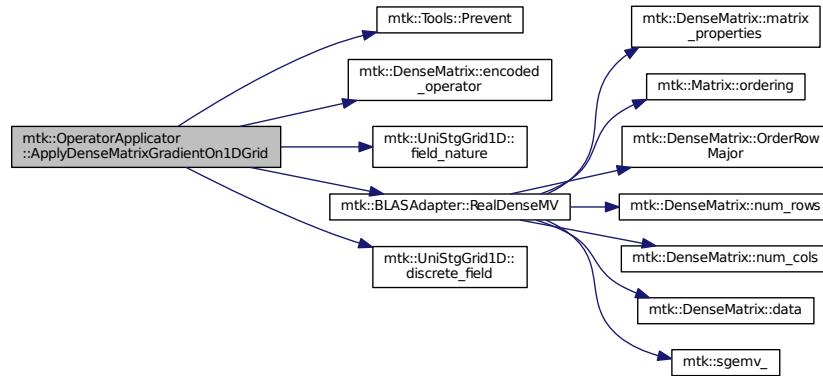
17.18.2.2 void mtk::OperatorApplicator::ApplyDenseMatrixGradientOn1DGrid (DenseMatrix & grad, UniStgGrid1D & grid, UniStgGrid1D & out) [static]

Parameters

in	<i>grad</i>	Gradient operator.
in	<i>grid</i>	1D grid.
out	<i>out</i>	Resulting 1D grid.

Definition at line 64 of file [mtk_operator_applicator.cc](#).

Here is the call graph for this function:



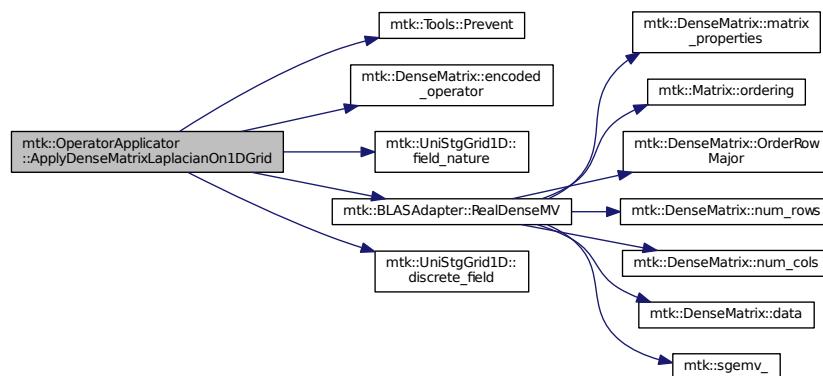
17.18.2.3 void mtk::OperatorApplicator::ApplyDenseMatrixLaplacianOn1DGrid (DenseMatrix & lap, UniStgGrid1D & grid, UniStgGrid1D & out) [static]

Parameters

in	<i>lap</i>	Laplacian operator.
in	<i>grid</i>	1D grid.
out	<i>out</i>	Resulting 1D grid.

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

Here is the call graph for this function:



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

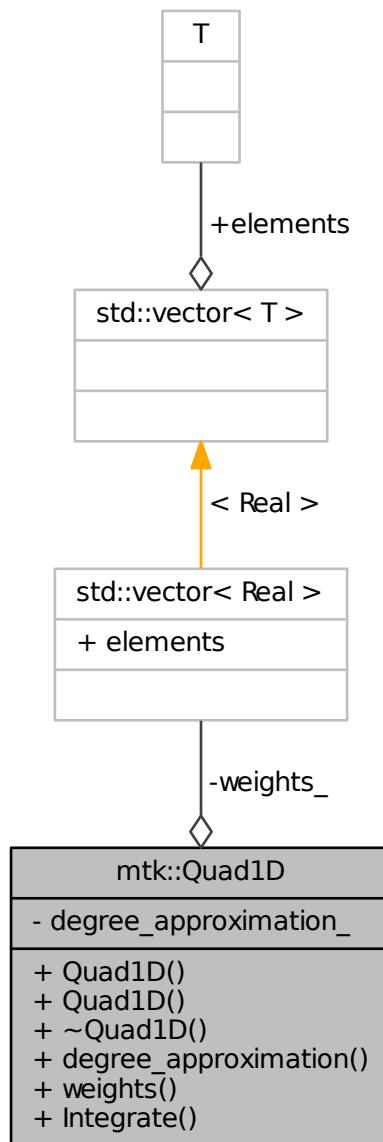
- [include/mtk_operator_applicator.h](#)
- [src/mtk_operator_applicator.cc](#)

17.19 mtk::Quad1D Class Reference

Implements a 1D mimetic quadrature.

```
#include <mtk_quad_1d.h>
```

Collaboration diagram for mtk::Quad1D:



Public Member Functions

- `Quad1D ()`
Default constructor.
- `Quad1D (const Quad1D &quad)`
Copy constructor.
- `~Quad1D ()`
Destructor.
- `int degree_approximation () const`
Get the degree of interpolating polynomial per sub-interval of domain.
- `Real * weights () const`
Return collection of weights.
- `Real Integrate (Real(*Integrand)(Real xx), UniStgGrid1D grid) const`
Mimetic integration routine.

Private Attributes

- `int degree_approximation_`
Degree of the interpolating polynomial.
- `std::vector< Real > weights_`
Collection of weights.

Friends

- `std::ostream & operator<< (std::ostream &stream, Quad1D &in)`
Output stream operator for printing.

17.19.1 Detailed Description

This class implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

Definition at line 81 of file [mtk_quad_1d.h](#).

17.19.2 Constructor & Destructor Documentation

17.19.2.1 `mtk::Quad1D::Quad1D ()`

17.19.2.2 `mtk::Quad1D::Quad1D (const Quad1D & quad)`

Parameters

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

17.19.2.3 `mtk::Quad1D::~Quad1D()`

17.19.3 Member Function Documentation

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

Returns

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

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

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.

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

Returns

Collection of weights.

17.19.4 Friends And Related Function Documentation

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

17.19.5 Member Data Documentation

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

Definition at line 130 of file [mtk_quad_1d.h](#).

17.19.5.2 `std::vector<Real> mtk::Quad1D::weights_ [private]`

Definition at line 132 of file [mtk_quad_1d.h](#).

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

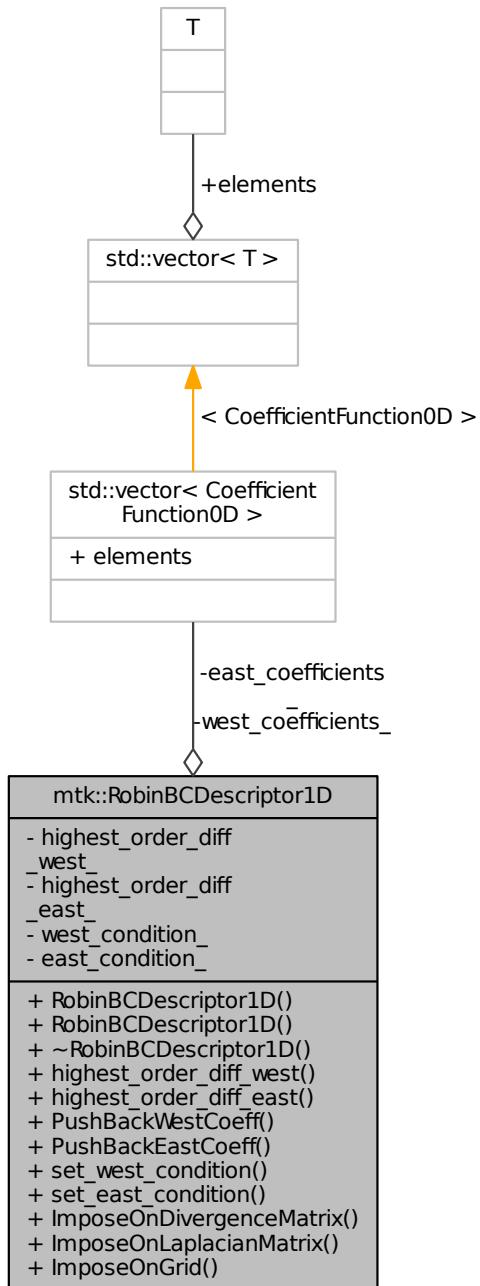
- [include/mtk_quad_1d.h](#)

17.20 mtk::RobinBCDescriptor1D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_1d.h>
```

Collaboration diagram for mtk::RobinBCDescriptor1D:



Public Member Functions

- [RobinBCDescriptor1D \(\)](#)

- *Default constructor.*
- `RobinBCDescriptor1D (const RobinBCDescriptor1D &desc)`
- Copy constructor.*
- `~RobinBCDescriptor1D () noexcept`
- Destructor.*
- `int highest_order_diff_west () const noexcept`

Getter for the highest order of differentiation in the west boundary.
- `int highest_order_diff_east () const noexcept`

Getter for the highest order of differentiation in the east boundary.
- `void PushBackWestCoeff (CoefficientFunction0D cw)`

Push back coefficient function at west of lowest order diff. available.
- `void PushBackEastCoeff (CoefficientFunction0D ce)`

Push back coefficient function at east of lowest order diff. available.
- `void set_west_condition (Real(*west_condition)(const Real &tt)) noexcept`

Set boundary condition at west.
- `void set_east_condition (Real(*east_condition)(const Real &tt)) noexcept`

Set boundary condition at east.
- `bool ImposeOnDivergenceMatrix (const Div1D &div, DenseMatrix &matrix, const std::vector< Real > ¶meters=std::vector< Real >(), const Real &time=mtk::kZero) const`

Imposes the condition on the operator represented as matrix.
- `bool ImposeOnLaplacianMatrix (const Lap1D &lap, DenseMatrix &matrix, const std::vector< Real > ¶meters=std::vector< Real >(), const Real &time=mtk::kZero) const`

Imposes the condition on the operator represented as matrix.
- `void ImposeOnGrid (UniStgGrid1D &grid, const Real &time=mtk::kZero) const`

Imposes the condition on the grid.

Private Attributes

- `int highest_order_diff_west_`

Highest order of differentiation west.
- `int highest_order_diff_east_`

Highest order of differentiation east.
- `std::vector< CoefficientFunction0D > west_coefficients_`

Coeffs. west.
- `std::vector< CoefficientFunction0D > east_coefficients_`

Coeffs. east.
- `Real(* west_condition_)(const Real &tt)`

Condition for west.
- `Real(* east_condition_)(const Real &tt)`

Condition for east.

17.20.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) = \beta_b(b, t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 153 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.2 Constructor & Destructor Documentation

17.20.2.1 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D()

Definition at line 93 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.20.2.2 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D(const RobinBCDescriptor1D & desc)

Parameters

in	desc	Given 1D descriptor.
----	------	----------------------

Definition at line 99 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.20.2.3 mtk::RobinBCDescriptor1D::~RobinBCDescriptor1D() [noexcept]

Definition at line 106 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.20.3 Member Function Documentation

17.20.3.1 int mtk::RobinBCDescriptor1D::highest_order_diff_east() const [noexcept]

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 113 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.20.3.2 int mtk::RobinBCDescriptor1D::highest_order_diff_east() const [noexcept]

Returns

Integer highest order of differentiation in the west boundary.

Definition at line 108 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.20.3.3 bool mtk::RobinBCDescriptor1D::ImposeOnDivergenceMatrix(const Div1D & div, mtk::DenseMatrix & matrix, const std::vector< Real > & parameters = std::vector<Real>(), const Real & time = mtk::kZero) const

Parameters

in	<i>div</i>	Operator in the Matrix .
in, out	<i>matrix</i>	Input Divergence operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

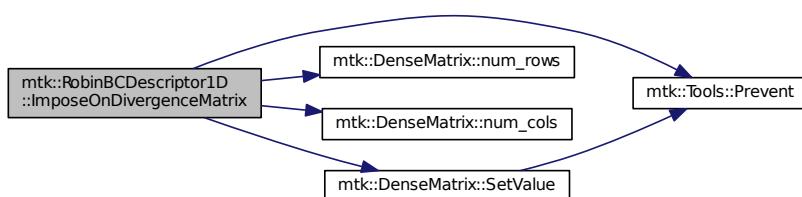
Returns

Success of the imposition.

1. Impose Dirichlet coefficients.
 - 1.1. Impose Dirichlet condition at the west.
 - 1.2. Impose Dirichlet condition at the east.
1. Impose Neumann coefficients.

Definition at line 166 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



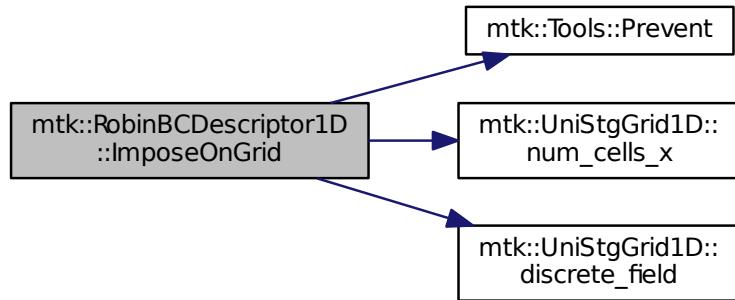
17.20.3.4 void mtk::RobinBCDescriptor1D::ImposeOnGrid(UniStgGrid1D & grid, const Real & time = mtk::kZero) const

Parameters

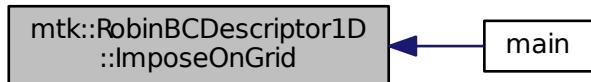
in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

Definition at line 285 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.3.5 `bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix (const Lap1D & lap, mtk::DenseMatrix & matrix, const std::vector< Real > & parameters = std::vector<Real>(), const Real & time = mtk::kZero) const`

Parameters

in	<i>lap</i>	Operator in the Matrix .
in, out	<i>matrix</i>	Input Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

Returns

Success of the imposition.

1. Impose Dirichlet coefficients.

- 1.1. Impose Dirichlet condition at the west.
- 1.2. Impose Dirichlet condition at the east.

1. Impose Neumann coefficients.

- 2.1. Create a mimetic gradient to approximate the first derivative.
- 2.2. Extract the coefficients approximating the boundary.

Warning

Coefficients returned by the mim_bndy getter are dimensionless! Therefore we must scale them by delta_x (from the grid), before adding to the matrix! But this information is in the given lap!

- 2.3. Impose Neumann condition at the west.
- 2.3.1. Get gradient coefficient and scale it.
- 2.3.2. Multiply times the coefficient for this boundary, times the unit normal for this boundary.
- 2.3.3. Set the final value summing it with what is on the matrix.
- 2.4. Impose Neumann condition at the east.

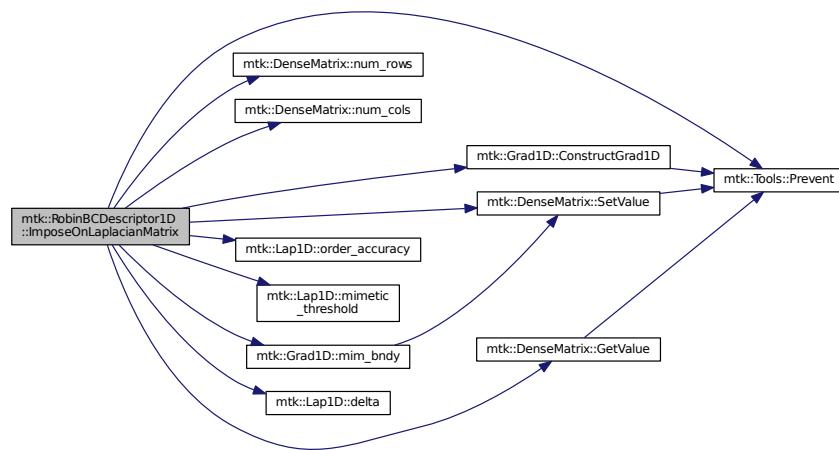
Warning

The Coefficients returned by the mim_bndy getter are those intended for the west boundary. We must enforce the center-skew-symmetry of the resulting operator by permuting their location in the matrix, and changing their sign.

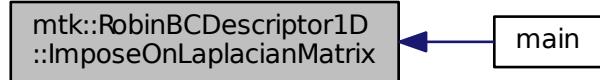
- 2.4.1. Get gradient coefficient and scale it.
- 2.4.2. Multiply times the coefficient for this boundary, times the unit normal for this boundary, and change the sign to enforce center-skew-symmetry.
- 2.4.3. Set the final value summing it with what is on the matrix.

Definition at line 200 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.3.6 void mtk::RobinBCDescriptor1D::PushBackEastCoeff (mtk::CoefficientFunction0D ce)

Parameters

in	ce	Function $c_e(x,y) : \Omega \mapsto \mathbb{R}$.
----	----	---

Definition at line 132 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.3.7 void mtk::RobinBCDescriptor1D::PushBackWestCoeff (mtk::CoefficientFunction0D cw)

Parameters

in	c_w	Function $c_w(x,y) : \Omega \mapsto \mathbb{R}$.
----	-------	---

Definition at line 118 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.3.8 void mtk::RobinBCDescriptor1D::set_east_condition (Real(*)(const Real &t) east_condition) [noexcept]

Parameters

in	$east_condition$	$\beta_e(y,t) : \Omega \mapsto \mathbb{R}$.
----	-------------------	--

Definition at line 156 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.3.9 void mtk::RobinBCDescriptor1D::set_west_condition (Real(*)(const Real &t) west_condition) [noexcept]

Parameters

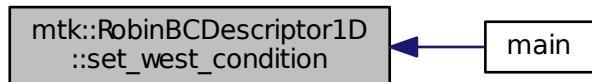
in	west_condition	$\beta_w(y, t) : \Omega \mapsto \mathbb{R}$.
----	----------------	---

Definition at line 146 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.20.4 Member Data Documentation

17.20.4.1 std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::east_coefficients_ [private]

Definition at line 256 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.4.2 `Real(* mtk::RobinBCDescriptor1D::east_condition_)(const Real &tt)` [private]

Definition at line 259 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.4.3 `int mtk::RobinBCDescriptor1D::highest_order_diff_east_` [private]

Definition at line 253 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.4.4 `int mtk::RobinBCDescriptor1D::highest_order_diff_west_` [private]

Definition at line 252 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.4.5 `std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::west_coefficients_` [private]

Definition at line 255 of file [mtk_robin_bc_descriptor_1d.h](#).

17.20.4.6 `Real(* mtk::RobinBCDescriptor1D::west_condition_)(const Real &tt)` [private]

Definition at line 258 of file [mtk_robin_bc_descriptor_1d.h](#).

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

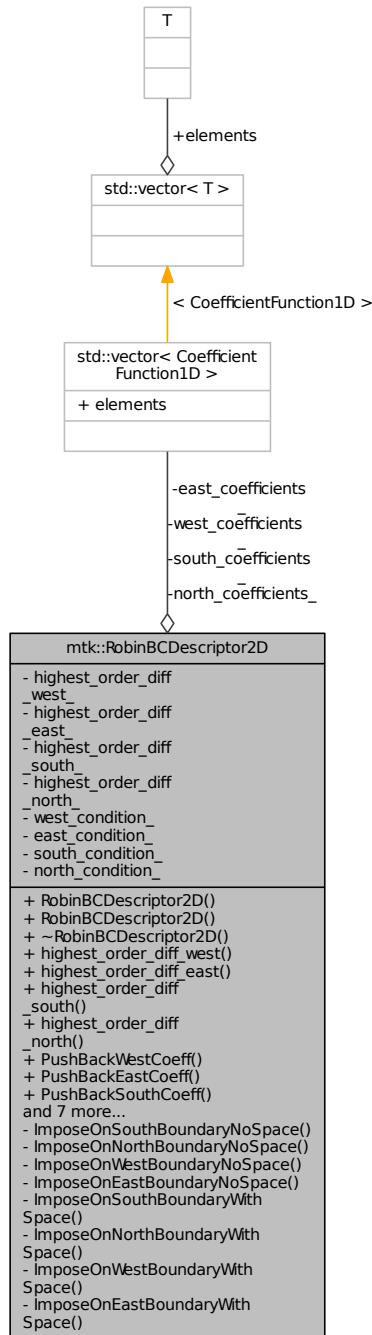
- [include/mtk_robin_bc_descriptor_1d.h](#)
- [src/mtk_robin_bc_descriptor_1d.cc](#)

17.21 mtk::RobinBCDescriptor2D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_2d.h>
```

Collaboration diagram for mtk::RobinBCDescriptor2D:



Public Member Functions

- [RobinBCDescriptor2D \(\)](#)

- Default constructor.*
- `RobinBCDescriptor2D (const RobinBCDescriptor2D &desc)`

Copy constructor.

 - `~RobinBCDescriptor2D () noexcept`

Destructor.

 - `int highest_order_diff_west () const noexcept`
Getter for the highest order of differentiation in the west boundary.
 - `int highest_order_diff_east () const noexcept`
Getter for the highest order of differentiation in the east boundary.
 - `int highest_order_diff_south () const noexcept`
Getter for the highest order of differentiation in the south boundary.
 - `int highest_order_diff_north () const noexcept`
Getter for the highest order of differentiation in the north boundary.
 - `void PushBackWestCoeff (CoefficientFunction1D cw)`
Push back coefficient function at west of lowest order diff. available.
 - `void PushBackEastCoeff (CoefficientFunction1D ce)`
Push back coefficient function at east of lowest order diff. available.
 - `void PushBackSouthCoeff (CoefficientFunction1D cs)`
Push back coefficient function south of lowest order diff. available.
 - `void PushBackNorthCoeff (CoefficientFunction1D cn)`
Push back coefficient function north of lowest order diff. available.
 - `void set_west_condition (Real(*west_condition)(const Real &yy, const Real &tt)) noexcept`
Set boundary condition at west.
 - `void set_east_condition (Real(*east_condition)(const Real &yy, const Real &tt)) noexcept`
Set boundary condition at east.
 - `void set_south_condition (Real(*south_condition)(const Real &xx, const Real &tt)) noexcept`
Set boundary condition at south.
 - `void set_north_condition (Real(*north_condition)(const Real &xx, const Real &tt)) noexcept`
Set boundary condition at north.
 - `bool ImposeOnLaplacianMatrix (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`
Imposes the condition on the operator represented as matrix.
 - `void ImposeOnGrid (UniStgGrid2D &grid, const Real &time=kZero) const`
Imposes the condition on the grid.

Private Member Functions

- `bool ImposeOnSouthBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`
Imposes the condition on the south boundary.
- `bool ImposeOnNorthBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`
Imposes the condition on the north boundary.
- `bool ImposeOnWestBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`
Imposes the condition on the west boundary.
- `bool ImposeOnEastBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

- `bool ImposeOnSouthBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the east boundary.
- `bool ImposeOnNorthBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the south boundary.
- `bool ImposeOnWestBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the north boundary.
- `bool ImposeOnEastBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the west boundary.
- `ImposeOnBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the east boundary.

Private Attributes

- `int highest_order_diff_west_`

Highest order of differentiation west.
- `int highest_order_diff_east_`

Highest order of differentiation east.
- `int highest_order_diff_south_`

Highest order differentiation for south.
- `int highest_order_diff_north_`

Highest order differentiation for north.
- `std::vector<<CoefficientFunction1D>> west_coefficients_`

Coeffs. west.
- `std::vector<<CoefficientFunction1D>> east_coefficients_`

Coeffs. east.
- `std::vector<<CoefficientFunction1D>> south_coefficients_`

Coeffs. south.
- `std::vector<<CoefficientFunction1D>> north_coefficients_`

Coeffs. north.
- `Real(* west_condition_)(const Real &xx, const Real &tt)`

Condition west.
- `Real(* east_condition_)(const Real &xx, const Real &tt)`

Condition east.
- `Real(* south_condition_)(const Real &yy, const Real &tt)`

Cond. south.
- `Real(* north_condition_)(const Real &yy, const Real &tt)`

Cond. north.

17.21.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 132 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.2 Constructor & Destructor Documentation

17.21.2.1 `mtk::RobinBCDescriptor2D::RobinBCDescriptor2D()`

Definition at line 84 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.21.2.2 `mtk::RobinBCDescriptor2D::RobinBCDescriptor2D(const RobinBCDescriptor2D & desc)`

Parameters

in	desc	Given 2D descriptor.
----	------	----------------------

Definition at line 94 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.21.2.3 `mtk::RobinBCDescriptor2D::~RobinBCDescriptor2D() [noexcept]`

Definition at line 105 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.21.3 Member Function Documentation

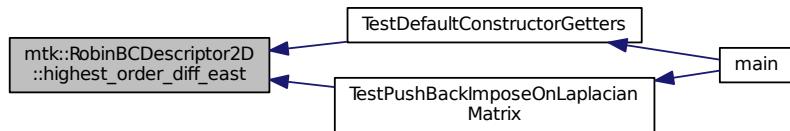
17.21.3.1 `int mtk::RobinBCDescriptor2D::highest_order_diff_east() const [noexcept]`

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 112 of file [mtk_robin_bc_descriptor_2d.cc](#).

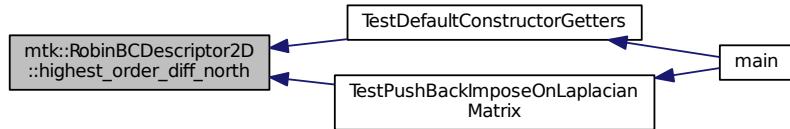
Here is the caller graph for this function:

**17.21.3.2 int mtk::RobinBCDescriptor2D::highest_order_diff_north () const [noexcept]****Returns**

Integer highest order of differentiation in the north boundary.

Definition at line 122 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the caller graph for this function:

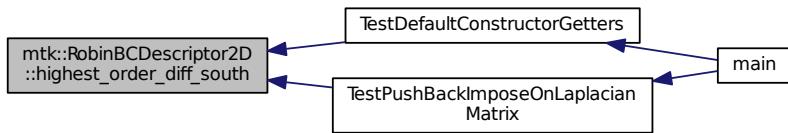
**17.21.3.3 int mtk::RobinBCDescriptor2D::highest_order_diff_south () const [noexcept]**

Returns

Integer highest order of differentiation in the south boundary.

Definition at line 117 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the caller graph for this function:



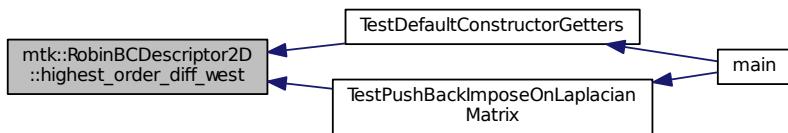
17.21.3.4 int mtk::RobinBCDescriptor2D::highest_order_diff_west () const [noexcept]

Returns

Integer highest order of differentiation in the west boundary.

Definition at line 107 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the caller graph for this function:



17.21.3.5 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]

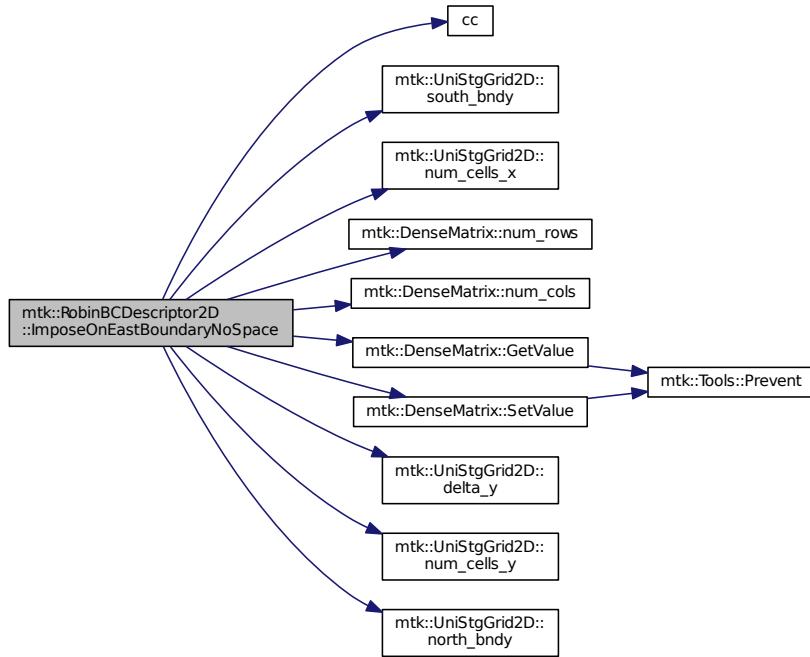
Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in,out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 495 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.6 `bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

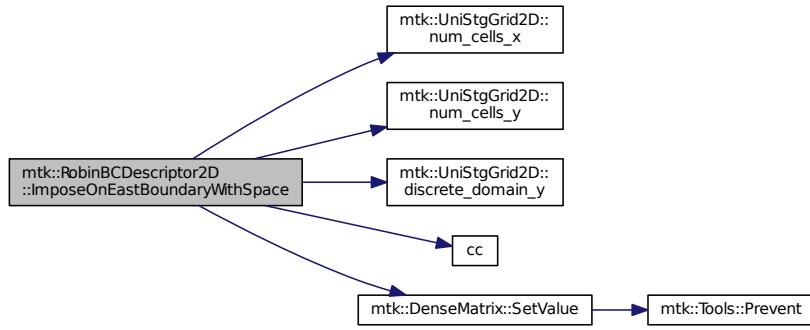
in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.

2. Impose the Neumann condition.

Definition at line 564 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.7 void mtk::RobinBCDescriptor2D::ImposeOnGrid (mtk::UniStgGrid2D & grid, const Real & time = kZero) const

Parameters

in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose assuming an scalar grid.

- 1.1. Impose south condition.

- 1.1.1. Impose south-west corner.

- 1.1.2. Impose south border.

- 1.1.3. Impose south-east corner.

- 1.2. Impose north condition.

- 1.2.1. Impose north-west corner.

- 1.2.2. Impose north border.

- 1.2.3. Impose north-east corner.

- 1.3. Impose west condition.

- 1.3.1. Impose south-west corner.

Note

As per discussion with Otilio, we will take the **arithmetic mean** of the values of the BCs at the corners.

- 1.3.2. Impose west border.

- 1.3.3. Impose north-west corner.

- 1.4. Impose east condition.

- 1.4.1. Impose south-east corner.

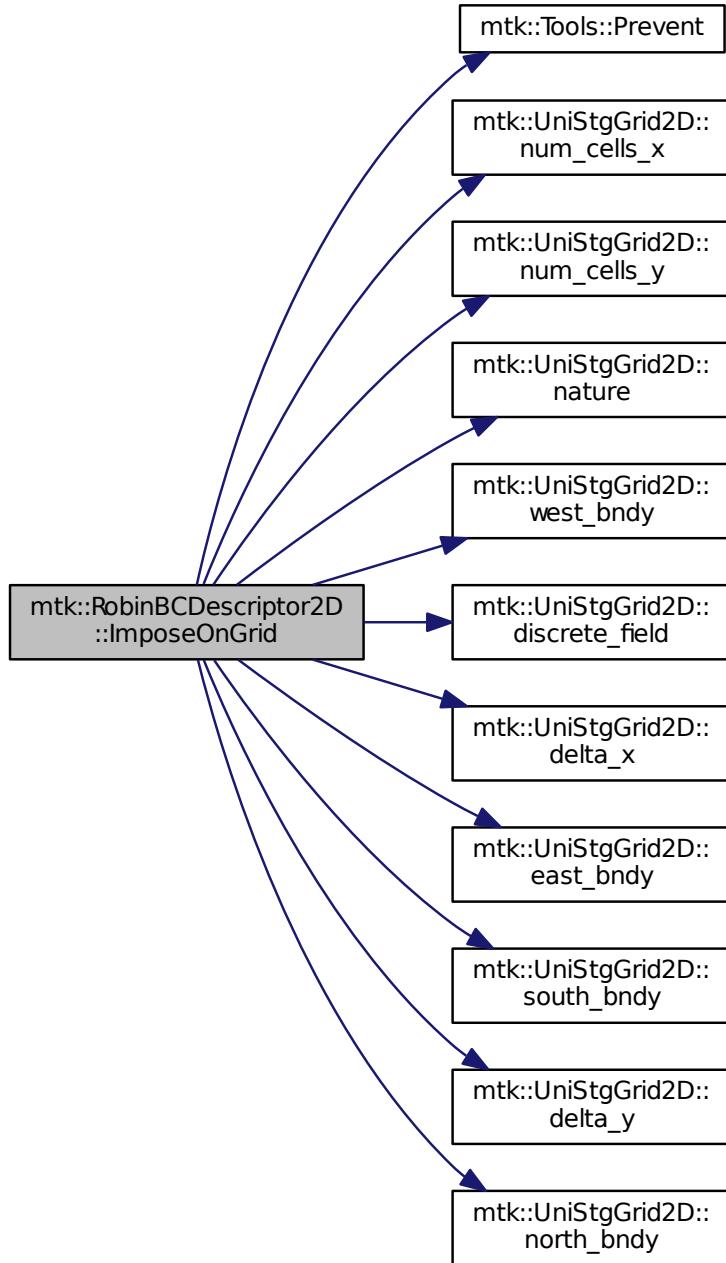
- 1.4.2. Impose east border.
- 1.4.3. Impose north-east corner.

- 1. Impose assuming a vector grid.

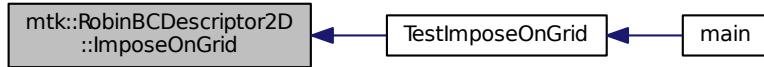
Todo Implement imposition for vector-valued grids. Need research here!

Definition at line 674 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.8 `bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix (const Lap2D & lap, const UniStgGrid2D & grid,
mtk::DenseMatrix & matrix, const Real & time = kZero) const`

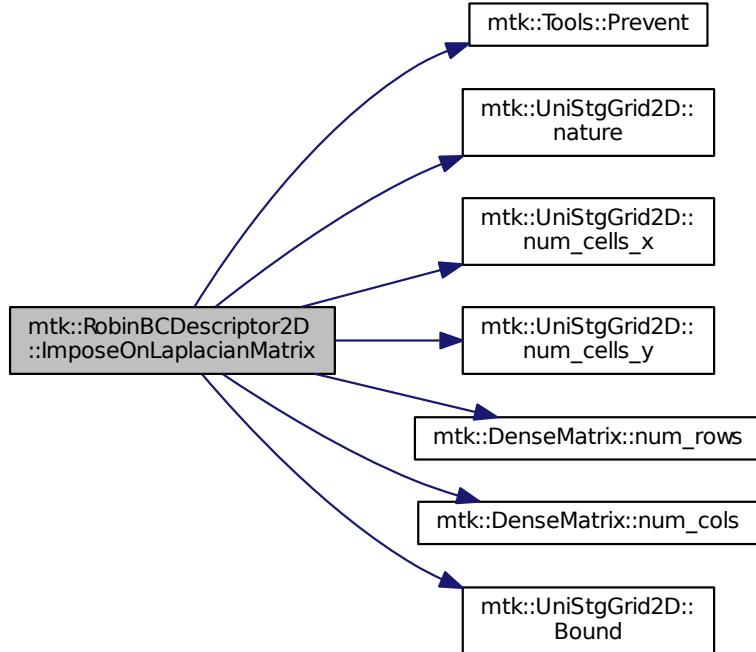
Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

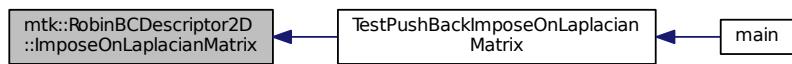
If we have not bound anything to the grid, then we have to generate our collection of spatial coordinates, as we evaluate the coefficients.

Definition at line 591 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.9 `bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

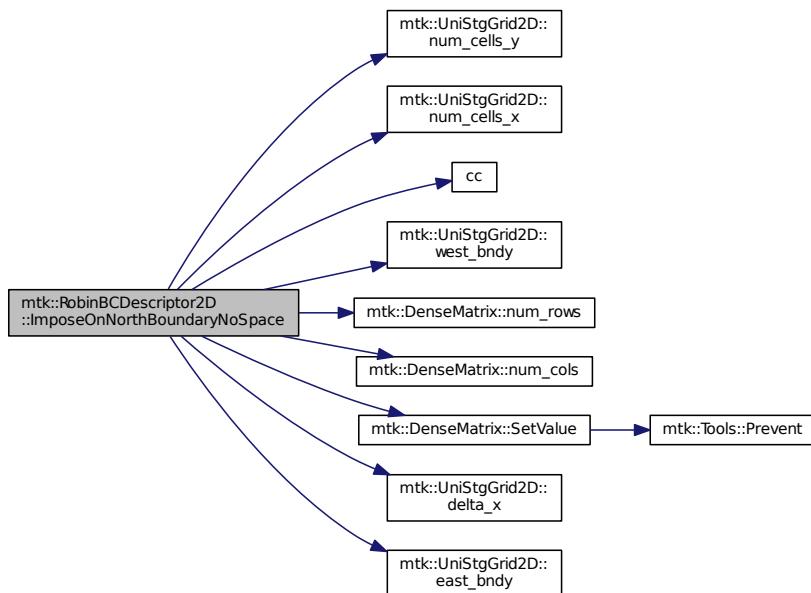
in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.

in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 312 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.10 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, mtk::DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose Dirichlet condition.

For each entry on the diagonal:

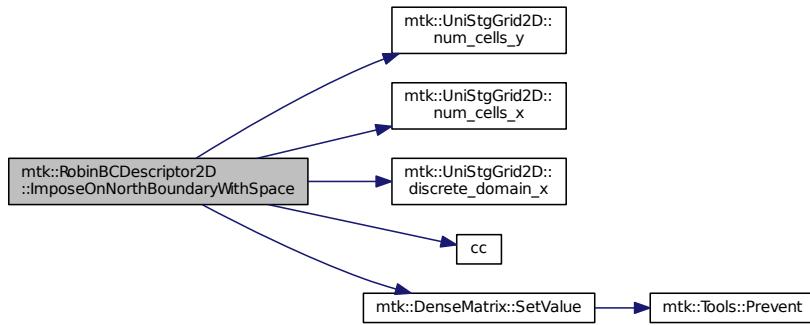
Evaluate next set spatial coordinates to evaluate the coefficient.

Evaluate and assign the Dirichlet coefficient.

1. Impose the Neumann condition.

Definition at line 372 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.11 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

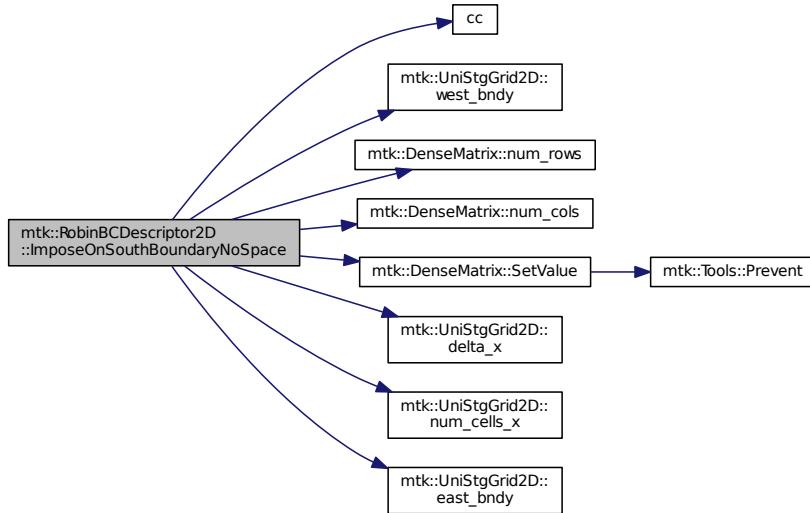
1. Impose the Dirichlet condition first.

2. Impose the Neumann condition.

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

Definition at line 229 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.12 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

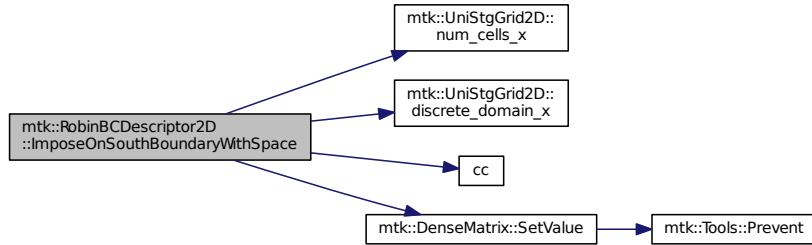
1. Impose the Dirichlet condition first.

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

1. Impose the Neumann condition.

Definition at line 284 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.13 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.

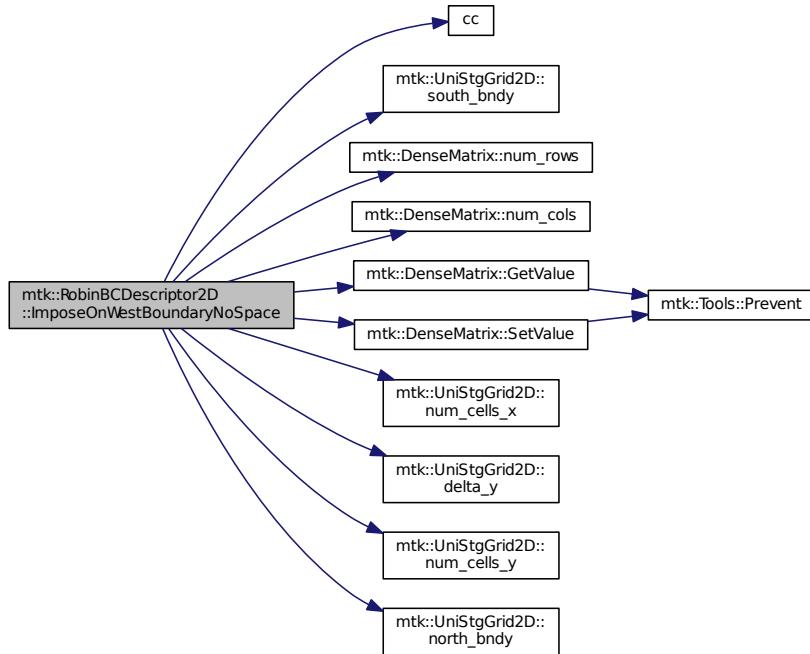
Note

As it can be seen, we must adopt a convention about how to treat the corners. Based on a reasoning with Otilio, we will take the **harmonic mean**.

1. Impose the Neumann condition.

Definition at line 399 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.14 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

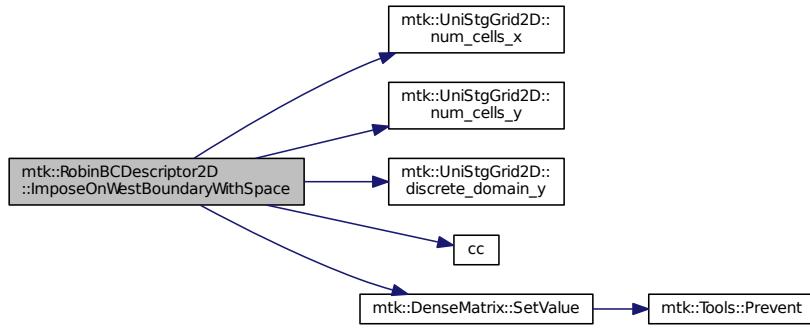
<code>in</code>	<code>lap</code>	Laplacian operator on the matrix.
<code>in</code>	<code>grid</code>	Grid upon which impose the desired boundary condition.
<code>in, out</code>	<code>matrix</code>	Input matrix with the Laplacian operator.
<code>in</code>	<code>time</code>	Current time snapshot. Default is kZero.

1. Impose the Dirichlet condition first.

2. Impose the Neumann condition.

Definition at line 468 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.21.3.15 void mtk::RobinBCDescriptor2D::PushBackEastCoeff (mtk::CoefficientFunction1D ce)

Parameters

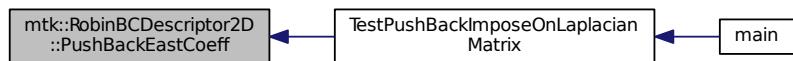
in	<i>cw</i>	Coeff. $c_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------	--

Definition at line 141 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.16 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff (mtk::CoefficientFunction1D cn)

Parameters

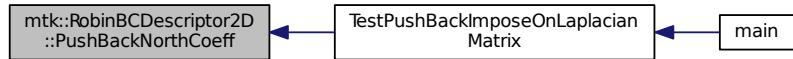
in	<i>cw</i>	Coeff. $c_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------	--

Definition at line 169 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.17 void mtk::RobinBCDescriptor2D::PushBackSouthCoeff (mtk::CoefficientFunction1D cs)

Parameters

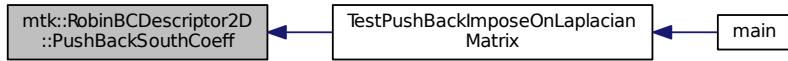
in	<i>cw</i>	Coeff. $c_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------	--

Definition at line 155 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.18 void mtk::RobinBCDescriptor2D::PushBackWestCoeff (mtk::CoefficientFunction1D cw)

Parameters

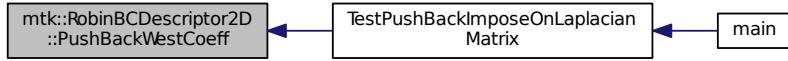
in	<i>cw</i>	Coeff. $c_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------	--

Definition at line 127 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.19 void mtk::RobinBCDescriptor2D::set_east_condition (Real(*)(const Real &y, const Real &t) east_condition) [noexcept]

Parameters

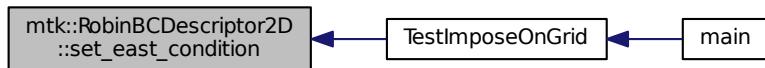
in	<i>east_condition</i>	$\beta_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------------------	---

Definition at line 194 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.20 `void mtk::RobinBCDescriptor2D::set_north_condition (Real(*)(const Real &xx, const Real &tt) north_condition) [noexcept]`

Parameters

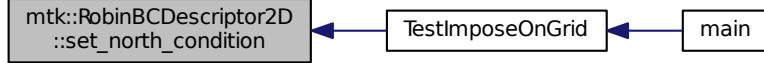
in	<i>north_condition</i>	$\beta_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	------------------------	---

Definition at line 217 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.21 void mtk::RobinBCDescriptor2D::set_south_condition (Real(*)(const Real &xx, const Real &tt) *south_condition*)
[noexcept]

Parameters

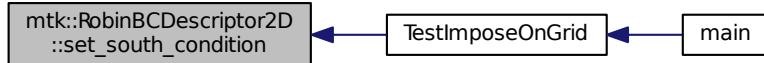
in	<i>south_condition</i>	$\beta_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	------------------------	---

Definition at line 205 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.3.22 void mtk::RobinBCDescriptor2D::set_west_condition (Real(*)(const Real &yy, const Real &tt) *west_condition*)
[noexcept]

Parameters

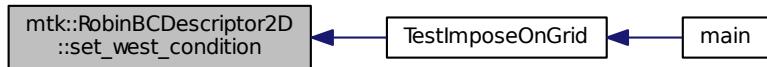
in	<i>west_condition</i> $\beta_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	---

Definition at line 183 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.21.4 Member Data Documentation

17.21.4.1 std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::east_coefficients_ [private]

Definition at line 371 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.2 Real(* mtk::RobinBCDescriptor2D::east_condition_)(const Real &xx, const Real &tt) [private]

Definition at line 376 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.3 int mtk::RobinBCDescriptor2D::highest_order_diff_east_ [private]

Definition at line 366 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.4 int mtk::RobinBCDescriptor2D::highest_order_diff_north_ [private]

Definition at line 368 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.5 int mtk::RobinBCDescriptor2D::highest_order_diff_south_ [private]

Definition at line 367 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.6 `int mtk::RobinBCDescriptor2D::highest_order_diff_west_ [private]`

Definition at line 365 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.7 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::north_coefficients_ [private]`

Definition at line 373 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.8 `Real(* mtk::RobinBCDescriptor2D::north_condition_)(const Real &yy, const Real &tt) [private]`

Definition at line 378 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.9 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::south_coefficients_ [private]`

Definition at line 372 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.10 `Real(* mtk::RobinBCDescriptor2D::south_condition_)(const Real &yy, const Real &tt) [private]`

Definition at line 377 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.11 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::west_coefficients_ [private]`

Definition at line 370 of file [mtk_robin_bc_descriptor_2d.h](#).

17.21.4.12 `Real(* mtk::RobinBCDescriptor2D::west_condition_)(const Real &xx, const Real &tt) [private]`

Definition at line 375 of file [mtk_robin_bc_descriptor_2d.h](#).

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

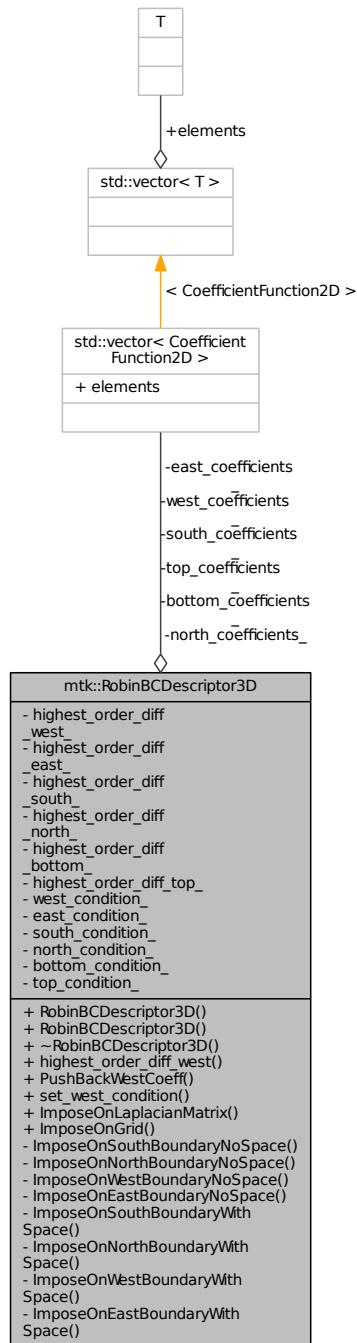
- include/[mtk_robin_bc_descriptor_2d.h](#)
- src/[mtk_robin_bc_descriptor_2d.cc](#)

17.22 mtk::RobinBCDescriptor3D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_3d.h>
```

Collaboration diagram for mtk::RobinBCDescriptor3D:



Public Member Functions

- [• RobinBCDescriptor3D \(\)](#)

- Default constructor.*
- `RobinBCDescriptor3D (const RobinBCDescriptor3D &desc)`

Copy constructor.

 - `~RobinBCDescriptor3D () noexcept`

Destructor.

 - int `highest_order_diff_west () const noexcept`

*Getter for highest order of differentiation in the * face.*

 - void `PushBackWestCoeff (CoefficientFunction2D cw)`

Push back coefficient function at west lowest order diff. available.

 - void `set_west_condition (Real(*west_condition)(const Real &xx, const Real &yy, const Real &tt)) noexcept`

Set boundary condition at west.

 - bool `ImposeOnLaplacianMatrix (const Lap3D &lap, const UniStgGrid3D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the operator represented as matrix.

 - void `ImposeOnGrid (UniStgGrid3D &grid, const Real &time=kZero) const`

Imposes the condition on the grid.

Private Member Functions

- bool `ImposeOnSouthBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the south boundary.

- bool `ImposeOnNorthBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the north boundary.

- bool `ImposeOnWestBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the west boundary.

- bool `ImposeOnEastBoundaryNoSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the east boundary.

- bool `ImposeOnSouthBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the south boundary.

- bool `ImposeOnNorthBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the north boundary.

- bool `ImposeOnWestBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the west boundary.

- bool `ImposeOnEastBoundaryWithSpace (const Lap2D &lap, const UniStgGrid2D &grid, DenseMatrix &matrix, const Real &time=kZero) const`

Imposes the condition on the east boundary.

Private Attributes

- int `highest_order_diff_west_`
Highest order of differentiation west.
- int `highest_order_diff_east_`
Highest order of differentiation east.
- int `highest_order_diff_south_`
Highest order differentiation for south.
- int `highest_order_diff_north_`
Highest order differentiation for north.
- int `highest_order_diff_bottom_`
Highest order differentiation bottom.
- int `highest_order_diff_top_`
Highest order differentiation for top.
- std::vector
 `< CoefficientFunction2D > west_coefficients_`
Coeffs. west.
- std::vector
 `< CoefficientFunction2D > east_coefficients_`
Coeffs. east.
- std::vector
 `< CoefficientFunction2D > south_coefficients_`
Coeffs. south.
- std::vector
 `< CoefficientFunction2D > north_coefficients_`
Coeffs. north.
- std::vector
 `< CoefficientFunction2D > bottom_coefficients_`
Coeffs. bottom.
- std::vector
 `< CoefficientFunction2D > top_coefficients_`
Coeffs. top.
- `Real(* west_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Condition west.
- `Real(* east_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Condition east.
- `Real(* south_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Cond. south.
- `Real(* north_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Cond. north.
- `Real(* bottom_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Cond. bottom.
- `Real(* top_condition_)(const Real &xx, const Real &yy, const Real &tt)`
Cond. top.

17.22.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 134 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.2 Constructor & Destructor Documentation

17.22.2.1 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D()`

17.22.2.2 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D(const RobinBCDescriptor3D & desc)`

Parameters

in	desc	Given 2D descriptor.
----	------	----------------------

17.22.2.3 `mtk::RobinBCDescriptor3D::~RobinBCDescriptor3D()` [noexcept]

17.22.3 Member Function Documentation

17.22.3.1 `int mtk::RobinBCDescriptor3D::highest_order_diff_west() const` [noexcept]

Returns

Integer highest order of differentiation in the * face.

17.22.3.2 `bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryNoSpace(const Lap2D & lap, const UniStgGrid2D & grid, DenseMatrix & matrix, const Real & time = kZero) const` [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.3 bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryWithSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.4 void mtk::RobinBCDescriptor3D::ImposeOnGrid (UniStgGrid3D & *grid*, const Real & *time* = kZero) const

Parameters

in, out	<i>grid</i>	Grid upon which impose the desired boundary condition.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.5 bool mtk::RobinBCDescriptor3D::ImposeOnLaplacianMatrix (const Lap3D & *lap*, const UniStgGrid3D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.6 bool mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryNoSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.7 bool mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryWithSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.8 bool mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryNoSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.9 bool mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryWithSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.10 bool mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryNoSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.
in	<i>time</i>	Current time snapshot. Default is kZero.

17.22.3.11 bool mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryWithSpace (const Lap2D & *lap*, const UniStgGrid2D & *grid*, DenseMatrix & *matrix*, const Real & *time* = kZero) const [private]

Parameters

in	<i>lap</i>	Laplacian operator on the matrix.
in	<i>grid</i>	Grid upon which impose the desired boundary condition.
in, out	<i>matrix</i>	Input matrix with the Laplacian operator.

in	<i>time</i>	Current time snapshot. Default is kZero.
----	-------------	--

17.22.3.12 void mtk::RobinBCDescriptor3D::PushBackWestCoeff (**CoefficientFunction2D** *cw*)

Parameters

in	<i>cw</i>	Coeff. $c_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------	---

17.22.3.13 void mtk::RobinBCDescriptor3D::set_west_condition (**Real(*)**(const Real &*xx*, const Real &*yy*, const Real &*tt*) *west_condition*) [noexcept]

Parameters

in	<i>west_condition</i>	$\beta_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.
----	-----------------------	--

17.22.4 Member Data Documentation

17.22.4.1 std::vector<**CoefficientFunction2D**> mtk::RobinBCDescriptor3D::bottom_coefficients_ [private]

Definition at line 307 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.2 **Real(*)** mtk::RobinBCDescriptor3D::bottom_condition_(const Real &*xx*, const Real &*yy*, const Real &*tt*) [private]

Definition at line 322 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.3 std::vector<**CoefficientFunction2D**> mtk::RobinBCDescriptor3D::east_coefficients_ [private]

Definition at line 304 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.4 **Real(*)** mtk::RobinBCDescriptor3D::east_condition_(const Real &*xx*, const Real &*yy*, const Real &*tt*) [private]

Definition at line 313 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.5 int mtk::RobinBCDescriptor3D::highest_order_diff_bottom_ [private]

Definition at line 300 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.6 int mtk::RobinBCDescriptor3D::highest_order_diff_east_ [private]

Definition at line 297 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.7 int mtk::RobinBCDescriptor3D::highest_order_diff_north_ [private]

Definition at line 299 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.8 `int mtk::RobinBCDescriptor3D::highest_order_diff_south_ [private]`

Definition at line 298 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.9 `int mtk::RobinBCDescriptor3D::highest_order_diff_top_ [private]`

Definition at line 301 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.10 `int mtk::RobinBCDescriptor3D::highest_order_diff_west_ [private]`

Definition at line 296 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.11 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::north_coefficients_ [private]`

Definition at line 306 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.12 `Real(* mtk::RobinBCDescriptor3D::north_condition_)(const Real &xx, const Real &yy, const Real &tt) [private]`

Definition at line 319 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.13 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::south_coefficients_ [private]`

Definition at line 305 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.14 `Real(* mtk::RobinBCDescriptor3D::south_condition_)(const Real &xx, const Real &yy, const Real &tt) [private]`

Definition at line 316 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.15 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::top_coefficients_ [private]`

Definition at line 308 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.16 `Real(* mtk::RobinBCDescriptor3D::top_condition_)(const Real &xx, const Real &yy, const Real &tt) [private]`

Definition at line 325 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.17 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::west_coefficients_ [private]`

Definition at line 303 of file [mtk_robin_bc_descriptor_3d.h](#).

17.22.4.18 `Real(* mtk::RobinBCDescriptor3D::west_condition_)(const Real &xx, const Real &yy, const Real &tt)`
`[private]`

Definition at line 310 of file [mtk_robin_bc_descriptor_3d.h](#).

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

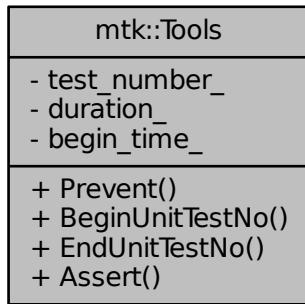
- [include/mtk_robin_bc_descriptor_3d.h](#)

17.23 mtk::Tools Class Reference

Tool manager class.

```
#include <mtk_tools.h>
```

Collaboration diagram for mtk::Tools:



Static Public Member Functions

- static void `Prevent` (const bool complement, const char *const fname, int lineno, const char *const fxname) noexcept

Enforces preconditions by preventing their complements from occur.
- static void `BeginUnitTestNo` (const int &n) noexcept

Begins the execution of a unit test. Starts a timer.
- static void `EndUnitTestNo` (const int &n) noexcept

Ends the execution of a unit test. Stops and reports wall-clock time.
- static void `Assert` (const bool &condition) noexcept

Asserts if the condition required to pass the unit test occurs.

Static Private Attributes

- static int `test_number_`

Current test being executed.

- static [Real duration_ {}](#)

Duration of the current test.

- static [clock_t begin_time_ {}](#)

Elapsed time on current test.

17.23.1 Detailed Description

Basic tools to ensure execution correctness, and to assists with unit testing.

Definition at line [76](#) of file [mtk_tools.h](#).

17.23.2 Member Function Documentation

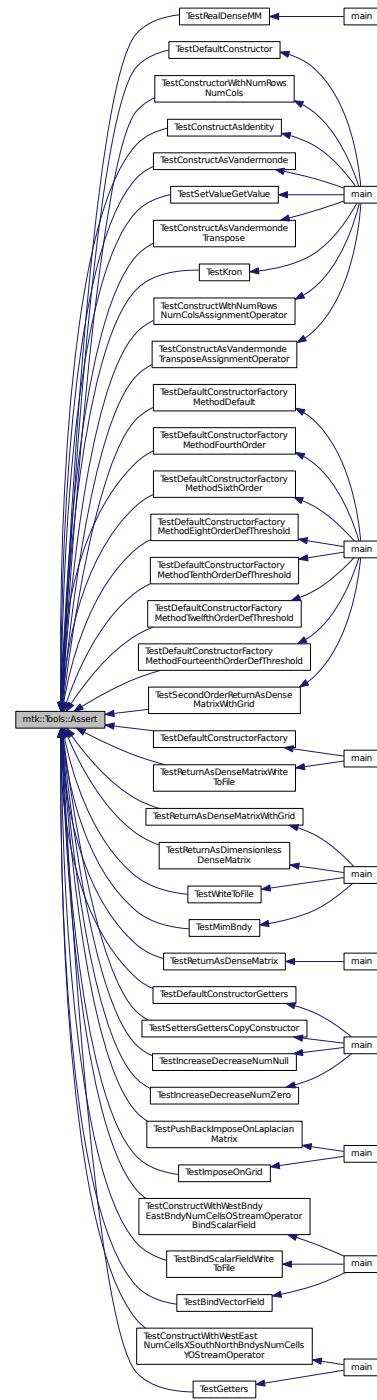
17.23.2.1 void mtk::Tools::Assert (const bool & condition) [static], [noexcept]

Parameters

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

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

Here is the caller graph for this function:



17.23.2.2 void mtk::Tools::BeginUnitTestNo (const int & nn) [static], [noexcept]

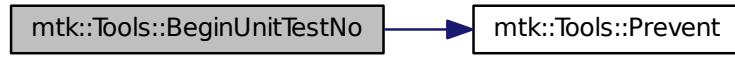
Parameters

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

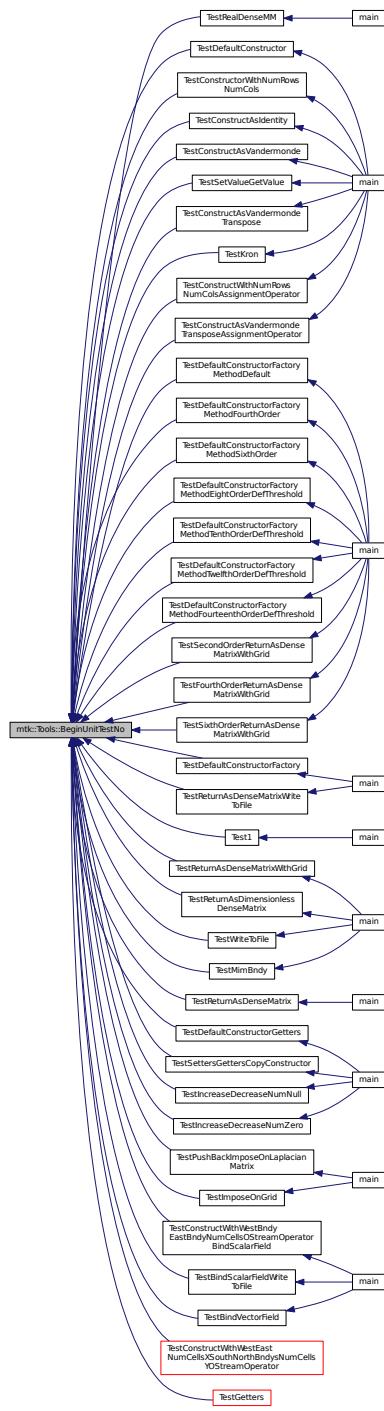
Todo Compute time using C++11 mechanisms.

Definition at line 101 of file [mtk_tools.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



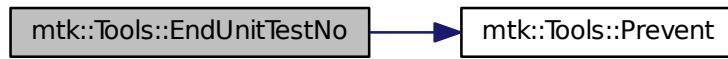
17.23.2.3 void mtk::Tools::EndUnitTestNo (const int & nn) [static], [noexcept]

Parameters

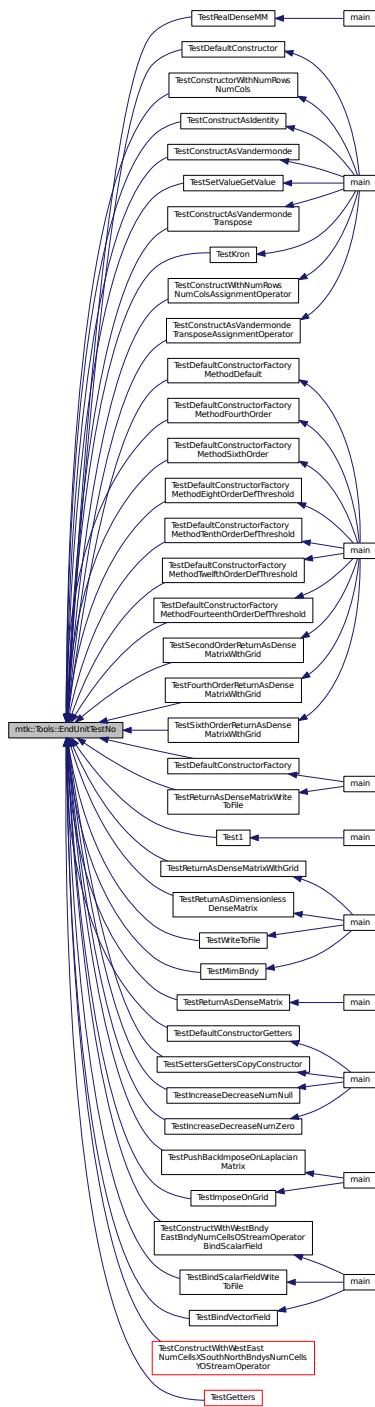
in	<i>nn</i>	Number of the test.
----	-----------	---------------------

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

See also

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

Parameters

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

Todo Check if this is the best way of stalling execution in C++11.

Definition at line 64 of file [mtk_tools.cc](#).

17.23.3 Member Data Documentation

17.23.3.1 `clock_t mtk::Tools::begin_time_ {} [static], [private]`

Definition at line 119 of file [mtk_tools.h](#).

17.23.3.2 `mtk::Real mtk::Tools::duration_ {} [static], [private]`

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

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

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

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

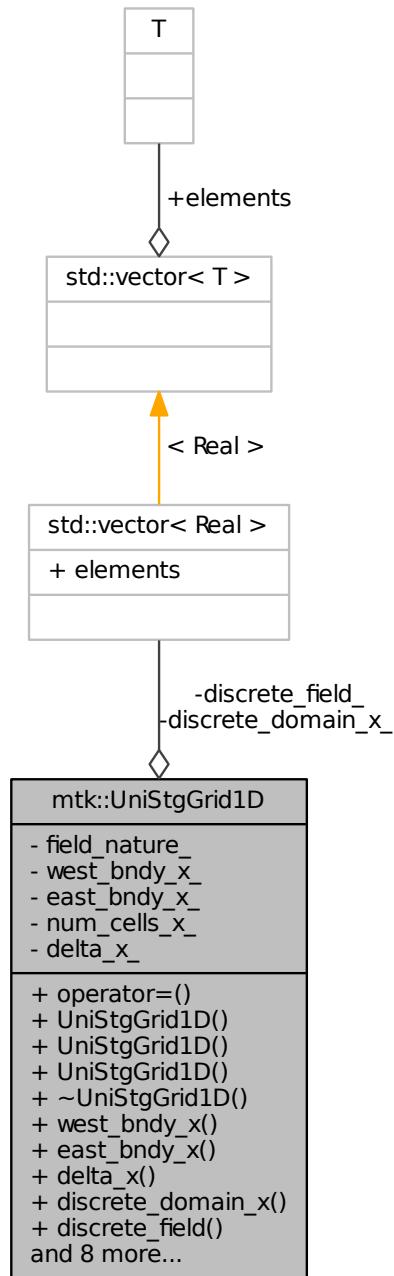
- [include/mtk_tools.h](#)
- [src/mtk_tools.cc](#)

17.24 mtk::UniStgGrid1D Class Reference

Uniform 1D Staggered Grid.

```
#include <mtk_uni_stg_grid_1d.h>
```

Collaboration diagram for mtk::UniStgGrid1D:



Public Member Functions

- `UniStgGrid1D & operator= (const UniStgGrid1D &in)`

- Overloaded assignment operator.*
- `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::FieldNature &field_nature=mtk::FieldNature::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 \$ x \$.
 - `const Real * discrete_domain_x () const`

Provides access to the grid spatial data.
 - `Real * discrete_field ()`

Provides access to the grid field data.
 - `int num_cells_x () const`

Provides access to the number of cells of the grid.
 - `FieldNature field_nature () const`

Provides access to the field nature.
 - `void GenerateDiscreteDomainX ()`

Generates the actual set of spatial coordinates.
 - `void ReserveDiscreteField ()`

Allocates memory for the discrete set of field samples.
 - `void BindScalarField (Real(*ScalarField)(const Real &xx, const std::vector< Real > &pp), const std::vector< Real > ¶meters=std::vector< Real >())`

Binds a given scalar field to the grid.
 - `void BindScalarField (const std::vector< Real > &samples)`

Binds a given scalar field (as an array) to the grid.
 - `void BindVectorField (Real(*VectorField)(const Real &xx, const std::vector< Real > &pp), const std::vector< Real > ¶meters=std::vector< Real >())`

Binds a given vector field to the grid.
 - `bool WriteToFile (std::string filename, std::string space_name, std::string field_name) const`

Writes grid to a file compatible with gnuplot 4.6.

Private Attributes

- `FieldNature field_nature_`

Nature of the discrete field.
- `std::vector< Real > discrete_domain_x_`

Array of spatial data.
- `std::vector< Real > discrete_field_`

Array of field's data.

- **Real west_bndy_x_**
West boundary spatial coordinate.
- **Real east_bndy_x_**
East boundary spatial coordinate.
- **Real num_cells_x_**
Number of cells discretizing the domain.
- **Real delta_x_**
Produced Δx .

Friends

- std::ostream & **operator<<** (std::ostream &stream, UniStgGrid1D &in)
Prints the grid as a tuple of arrays.

17.24.1 Detailed Description

Uniform 1D Staggered Grid.

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

17.24.2 Constructor & Destructor Documentation

17.24.2.1 mtk::UniStgGrid1D::UniStgGrid1D()

Definition at line 126 of file [mtk_uni_stg_grid_1d.cc](#).

17.24.2.2 mtk::UniStgGrid1D::UniStgGrid1D(const UniStgGrid1D & grid)

Parameters

in	grid	Given grid.
----	------	-------------

Definition at line 135 of file [mtk_uni_stg_grid_1d.cc](#).

17.24.2.3 mtk::UniStgGrid1D::UniStgGrid1D(const Real & west_bndy_x, const Real & east_bndy_x, const int & num_cells_x, const mtk::FieldNature & field_nature = mtk::FieldNature::SCALAR)

Parameters

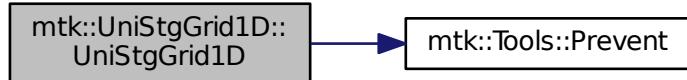
in	west_bndy_x	Coordinate for the west boundary.
in	east_bndy_x	Coordinate for the east boundary.
in	num_cells_x	Number of cells of the required grid.
in	field_nature	Nature of the discrete field to hold.

See also

[mtk::FieldNature](#)

Definition at line 151 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



17.24.2.4 mtk::UniStgGrid1D::~UniStgGrid1D()

Definition at line 171 of file [mtk_uni_stg_grid_1d.cc](#).

17.24.3 Member Function Documentation

17.24.3.1 void mtk::UniStgGrid1D::BindScalarField (Real(*)(const Real &xx, const std::vector< Real > &pp) *ScalarField*, const std::vector< Real > & *parameters* = std::vector<Real>())

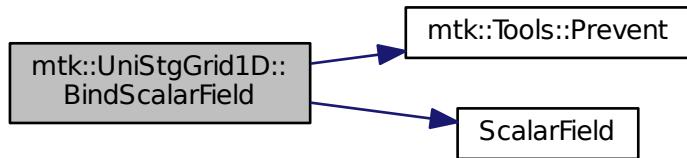
Parameters

in	<i>ScalarField</i>	Pointer to the function implementing the scalar field.
in	<i>parameters</i>	Array of parameters for the field to be evaluated.

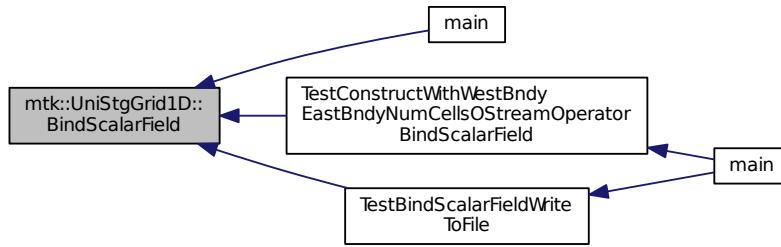
1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 260 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



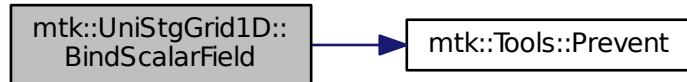
17.24.3.2 void mtk::UniStgGrid1D::BindScalarField (const std::vector< Real > & samples)

Parameters

in	<i>samples</i>	Array of samples.
----	----------------	-------------------

Definition at line 302 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



17.24.3.3 void mtk::UniStgGrid1D::BindVectorField (Real(*)(const Real &x, const std::vector< Real > &p) VectorField, const std::vector< Real > & parameters = std::vector<Real>())

We assume the field to be of the form:

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

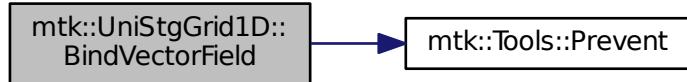
Parameters

in	<i>VectorField</i>	Pointer to the function implementing the vector field.
in	<i>parameters</i>	Array of parameters for the field to be evaluated.

1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 314 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



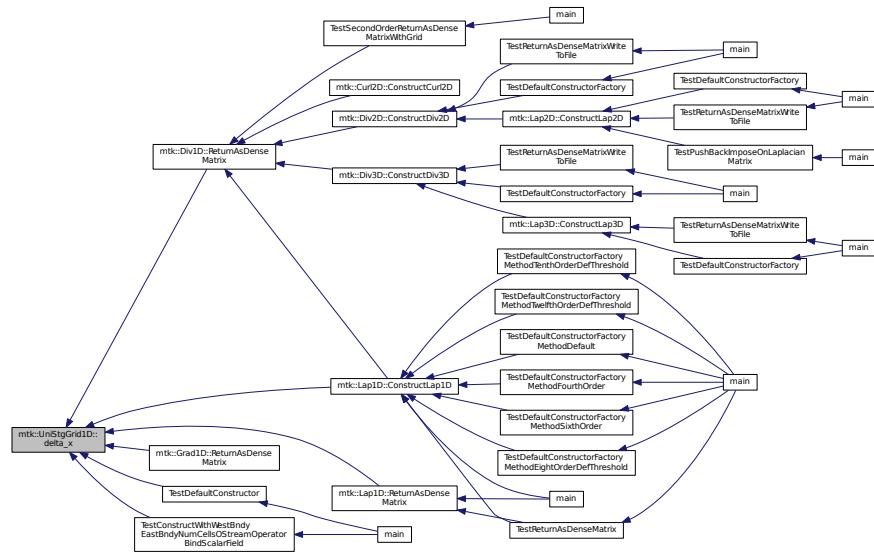
17.24.3.4 mtk::Real mtk::UniStgGrid1D::delta_x() const

Returns

Computed \$x\$.

Definition at line 183 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



17.24.3.5 const mtk::Real * mtk::UniStgGrid1D::discrete_domain_x() const

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

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

17.24.3.6 mtk::Real * mtk::UniStgGrid1D::discrete_field()

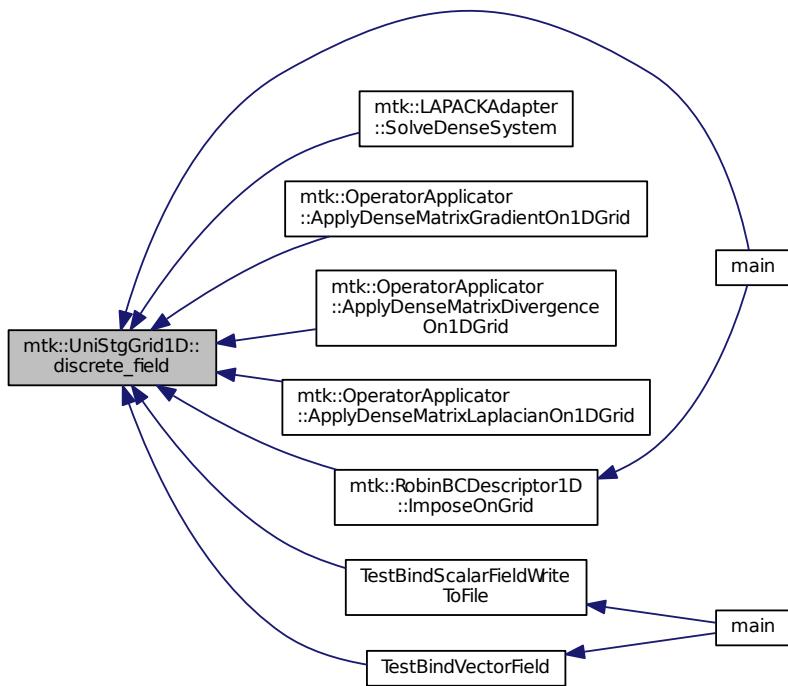
Returns

Pointer to the field data.

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

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

Here is the caller graph for this function:



17.24.3.7 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const

Returns

East boundary spatial coordinate.

Definition at line 178 of file [mtk_uni_stg_grid_1d.cc](#).

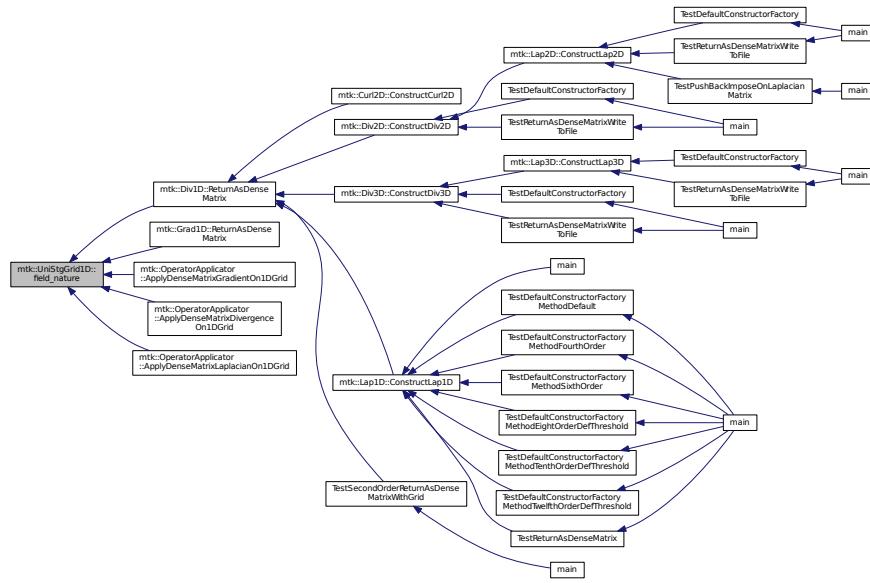
17.24.3.8 mtk::FieldNature mtk::UniStgGrid1D::field_nature() const

Returns

Nature of the filed on the grid.

Definition at line 203 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



17.24.3.9 void mtk::UniStgGrid1D::GenerateDiscreteDomainX ()

Definition at line 208 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



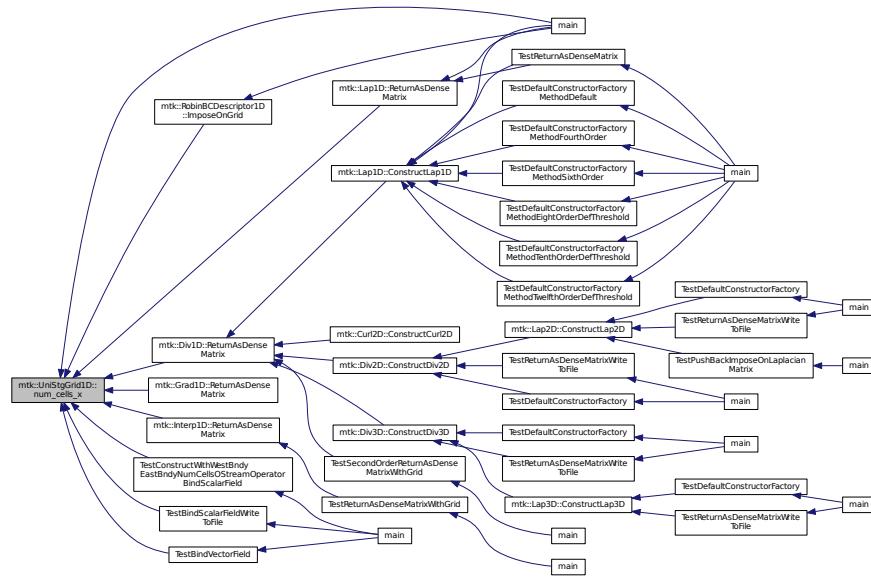
17.24.3.10 int mtk::UniStgGrid1D::num_cells_x () const

Returns

Number of cells of the grid.

Definition at line 198 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



17.24.3.11 `mtk::UniStgGrid1D & mtk::UniStgGrid1D::operator= (const UniStgGrid1D & in)`

Parameters

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

Returns

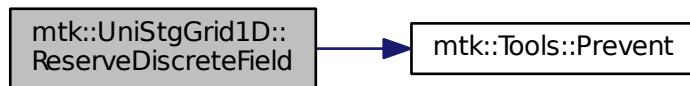
Copy of the given grid.

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

17.24.3.12 void mtk::UniStgGrid1D::ReserveDiscreteField()

Definition at line 243 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:

**17.24.3.13 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const****Returns**

West boundary spatial coordinate.

Definition at line 173 of file [mtk_uni_stg_grid_1d.cc](#).

17.24.3.14 bool mtk::UniStgGrid1D::WriteToFile(std::string filename, std::string space_name, std::string field_name) const**Parameters**

in	<i>filename</i>	Name of the output file.
in	<i>space_name</i>	Name for the first column of the data.
in	<i>field_name</i>	Name for the second column of the data.

Returns

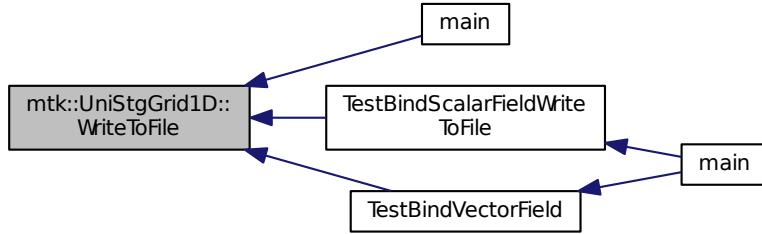
Success of the file writing process.

See also

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

Definition at line 348 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:



17.24.4 Friends And Related Function Documentation

17.24.4.1 `std::ostream& operator<< (std::ostream & stream, mtk::UniStgGrid1D & in)` [friend]

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

Definition at line 68 of file [mtk_uni_stg_grid_1d.cc](#).

17.24.5 Member Data Documentation

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

Definition at line 244 of file [mtk_uni_stg_grid_1d.h](#).

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

Definition at line 238 of file [mtk_uni_stg_grid_1d.h](#).

17.24.5.3 `std::vector<Real> mtk::UniStgGrid1D::discrete_field_` [private]

Definition at line 239 of file [mtk_uni_stg_grid_1d.h](#).

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

Definition at line 242 of file [mtk_uni_stg_grid_1d.h](#).

17.24.5.5 `FieldNature mtk::UniStgGrid1D::field_nature_` [private]

Definition at line 236 of file [mtk_uni_stg_grid_1d.h](#).

17.24.5.6 Real mtk::UniStgGrid1D::num_cells_x_ [private]

Definition at line 243 of file [mtk_uni_stg_grid_1d.h](#).

17.24.5.7 Real mtk::UniStgGrid1D::west_bndy_x_ [private]

Definition at line 241 of file [mtk_uni_stg_grid_1d.h](#).

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

- include/[mtk_uni_stg_grid_1d.h](#)

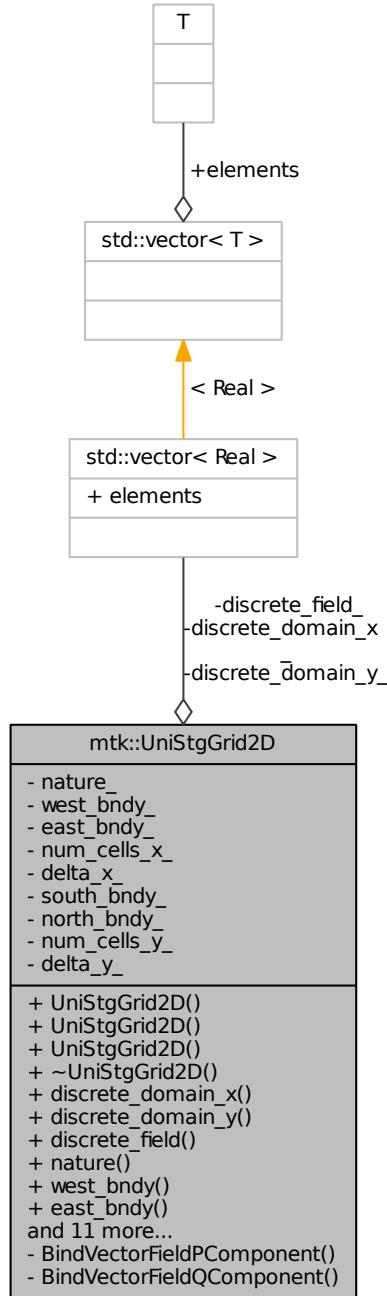
- src/[mtk_uni_stg_grid_1d.cc](#)

17.25 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::FieldNature::SCALAR)`

Construct a grid based on spatial discretization parameters.

- `~UniStgGrid2D ()`

Destructor.

- `const Real * discrete_domain_x () const`

Provides access to the grid spatial data.

- `const Real * discrete_domain_y () const`

Provides access to the grid spatial data.

- `Real * discrete_field ()`

Provides access to the grid field data.

- `FieldNature nature () const`

Physical nature of the data bound to the grid.

- `Real west_bndy () const`

Provides access to west boundary spatial coordinate.

- `Real east_bndy () const`

Provides access to east boundary spatial coordinate.

- `int num_cells_x () const`

Provides access to the number of cells of the grid.

- `Real delta_x () const`

Provides access to the computed \$x\$.

- `Real south_bndy () const`

Provides access to south boundary spatial coordinate.

- `Real north_bndy () const`

Provides access to north boundary spatial coordinate.

- `int num_cells_y () const`

Provides access to the number of cells of the grid.

- `Real delta_y () const`

Provides access to the computed \$y\$.

- `bool Bound () const`

Have any field been bound to the grid?

- `int Size () const`

Total number of samples in the grid.

- `void BindScalarField (Real(*ScalarField)(const Real &xx, const Real &yy))`

Binds a given scalar field to the grid.

- `void BindVectorField (Real(*VectorFieldPComponent)(const Real &xx, const Real &yy), Real(*VectorFieldQComponent)(const Real &xx, const Real &yy))`

Binds a given vector field to the grid.

- `bool WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string field_name) const`

Writes grid to a file compatible with Gnuplot 4.6.

Private Member Functions

- void `BindVectorFieldPComponent (Real(*VectorFieldPComponent)(const Real &xx, const Real &yy))`
Binds a given component of a vector field to the grid.
- void `BindVectorFieldQComponent (Real(*VectorFieldQComponent)(const Real &xx, const Real &yy))`
Binds a given component of a vector field to the grid.

Private Attributes

- `std::vector< Real > discrete_domain_x_`
Array of spatial data.
- `std::vector< Real > discrete_domain_y_`
Array of spatial data.
- `std::vector< Real > discrete_field_`
Array of field's data.
- `FieldNature nature_`
Nature of the discrete field.
- `Real west_bndy_`
West boundary spatial coordinate.
- `Real east_bndy_`
East boundary spatial coordinate.
- `int num_cells_x_`
Number of cells discretizing the domain.
- `Real delta_x_`
Computed Δx .
- `Real south_bndy_`
West boundary spatial coordinate.
- `Real north_bndy_`
East boundary spatial coordinate.
- `int num_cells_y_`
Number of cells discretizing the domain.
- `Real delta_y_`
Computed Δy .

Friends

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

17.25.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 80 of file `mtk_uni_stg_grid_2d.h`.

17.25.2 Constructor & Destructor Documentation

17.25.2.1 mtk::UniStgGrid2D::UniStgGrid2D()

Definition at line 132 of file [mtk_uni_stg_grid_2d.cc](#).

17.25.2.2 mtk::UniStgGrid2D::UniStgGrid2D(const UniStgGrid2D & grid)

Parameters

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

Definition at line 146 of file [mtk_uni_stg_grid_2d.cc](#).

17.25.2.3 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::FieldNature::SCALAR)

Parameters

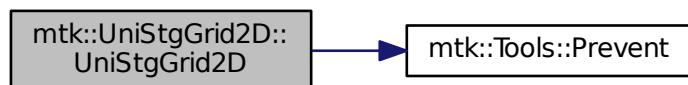
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</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 170 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



17.25.2.4 mtk::UniStgGrid2D::~UniStgGrid2D()

Definition at line 204 of file [mtk_uni_stg_grid_2d.cc](#).

17.25.3 Member Function Documentation

17.25.3.1 void mtk::UniStgGrid2D::BindScalarField (Real(*)(const Real &xx, const Real &yy) *ScalarField*)

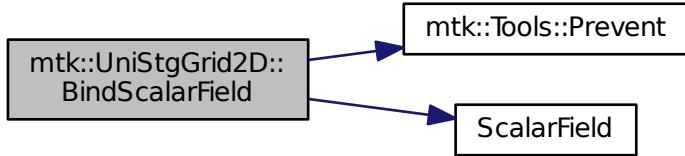
Parameters

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

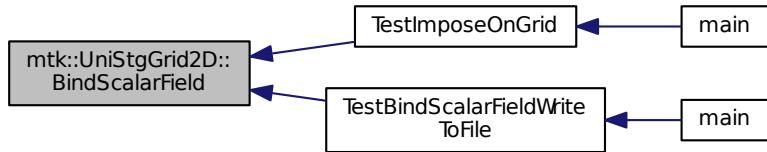
1. Create collection of spatial coordinates for x .
2. Create collection of spatial coordinates for y .
3. Create collection of field samples.

Definition at line 276 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.25.3.2 void mtk::UniStgGrid2D::BindVectorField (Real(*)(const Real &xx, const Real &yy) *VectorFieldPComponent*,
Real(*)(const Real &xx, const Real &yy) *VectorFieldQComponent*)

We assume the field to be of the form:

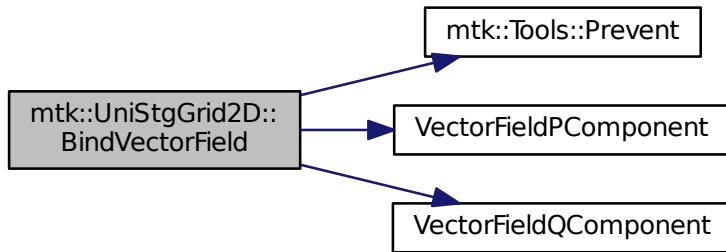
$$\mathbf{v}(\mathbf{x}) = p(x,y)\hat{\mathbf{i}} + q(x,y)\hat{\mathbf{j}}$$

Parameters

in	<i>VectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
in	<i>VectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.

Definition at line 425 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

We assume the field to be of the form:

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

Parameters

in	<i>BindVectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
----	----------------------------------	---

1. Create collection of spatial coordinates for x .
2. Create collection of spatial coordinates for y .

3. Allocate space for discrete vector field and bind \$ p \$ component.

Definition at line 332 of file [mtk_uni_stg_grid_2d.cc](#).

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

We assume the field to be of the form:

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

Parameters

in	<i>BindVectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
----	----------------------------------	---

1. Bind \$ q \$ component, since \$ p \$ component has already been bound.

Definition at line 397 of file [mtk_uni_stg_grid_2d.cc](#).

17.25.3.5 bool mtk::UniStgGrid2D::Bound () const

Returns

True is a field has been bound.

Definition at line 256 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



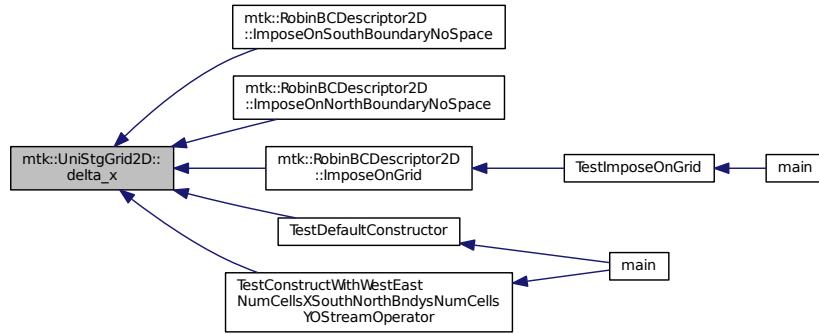
17.25.3.6 mtk::Real mtk::UniStgGrid2D::delta_x () const

Returns

Computed $\$ \times \$$.

Definition at line 226 of file [mtk_uni_stg_grid_2d.cc](#).

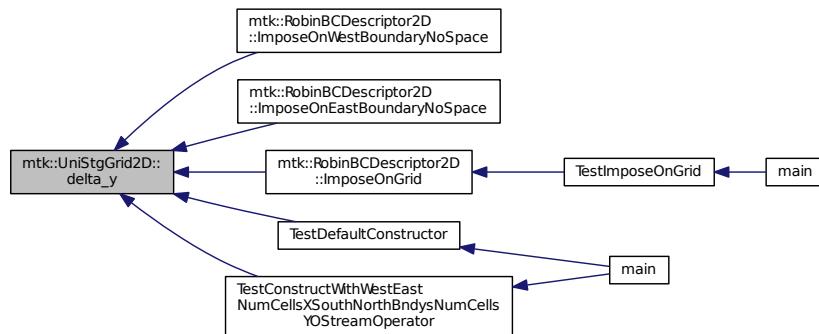
Here is the caller graph for this function:

**17.25.3.7 mtk::Real mtk::UniStgGrid2D::delta_y () const****Returns**

Computed $\$ y \$$.

Definition at line 251 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:

**17.25.3.8 const mtk::Real * mtk::UniStgGrid2D::discrete_domain_x () const**

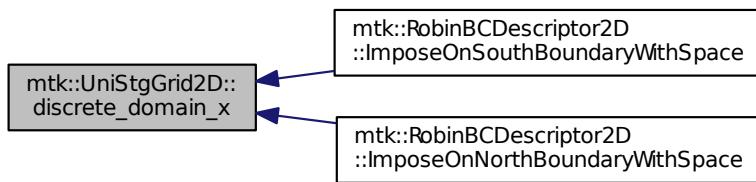
Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 231 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.25.3.9 const mtk::Real * mtk::UniStgGrid2D::discrete_domain_y () const

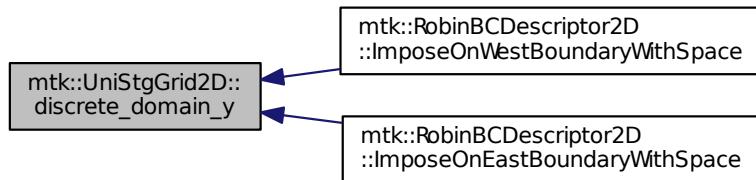
Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 261 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



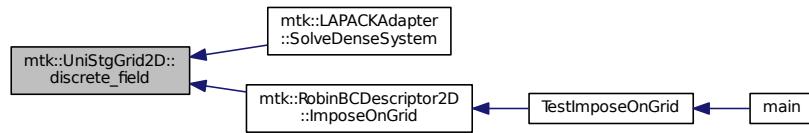
17.25.3.10 mtk::Real * mtk::UniStgGrid2D::discrete_field ()

Returns

Pointer to the field data.

Definition at line 266 of file [mtk_uni_stg_grid_2d.cc](#).

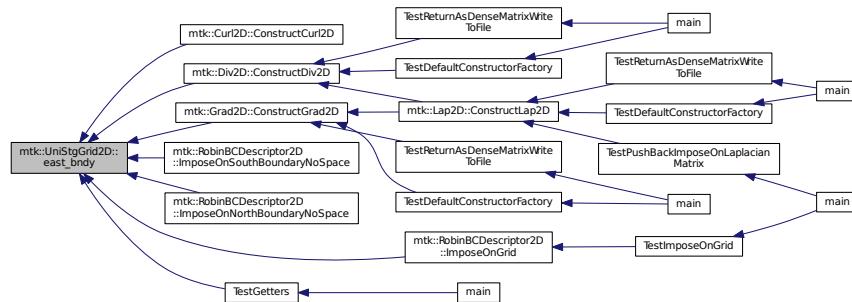
Here is the caller graph for this function:

**17.25.3.11 mtk::Real mtk::UniStgGrid2D::east_bndy () const****Returns**

East boundary spatial coordinate.

Definition at line 216 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:

**17.25.3.12 mtk::FieldNature mtk::UniStgGrid2D::nature () const****Returns**

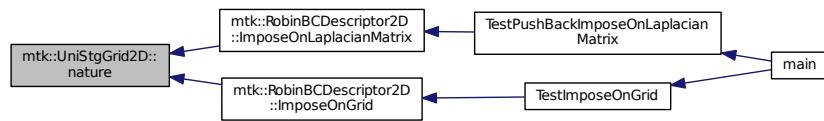
Value of an enumeration.

See also

[mtk::FieldNature](#)

Definition at line 206 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



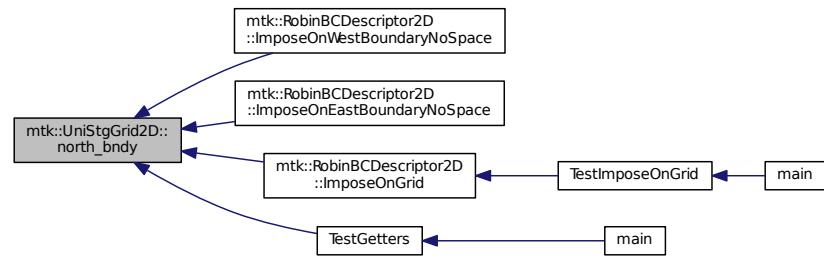
17.25.3.13 mtk::Real mtk::UniStgGrid2D::north_bndy() const

Returns

North boundary spatial coordinate.

Definition at line 241 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



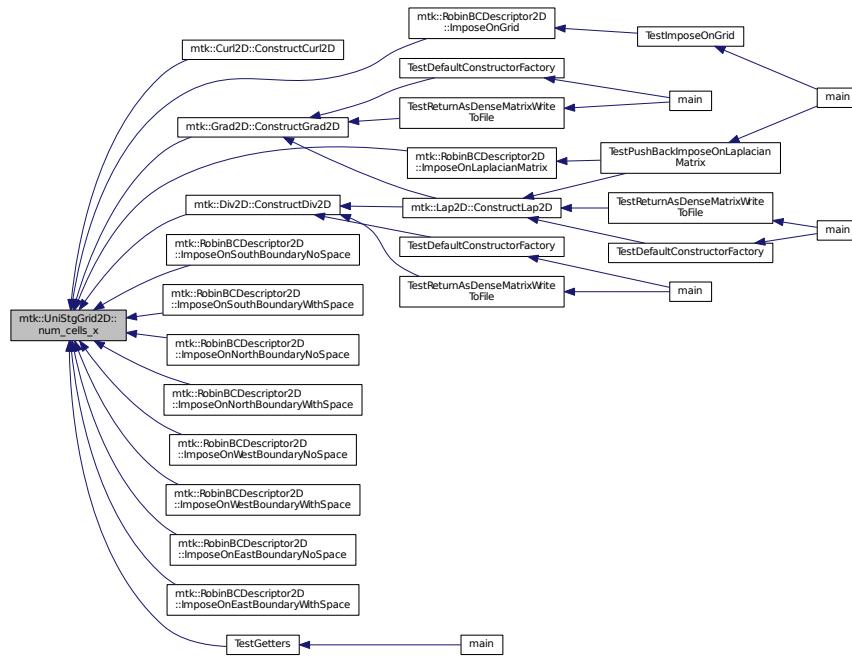
17.25.3.14 int mtk::UniStgGrid2D::num_cells_x() const

Returns

Number of cells of the grid.

Definition at line 221 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



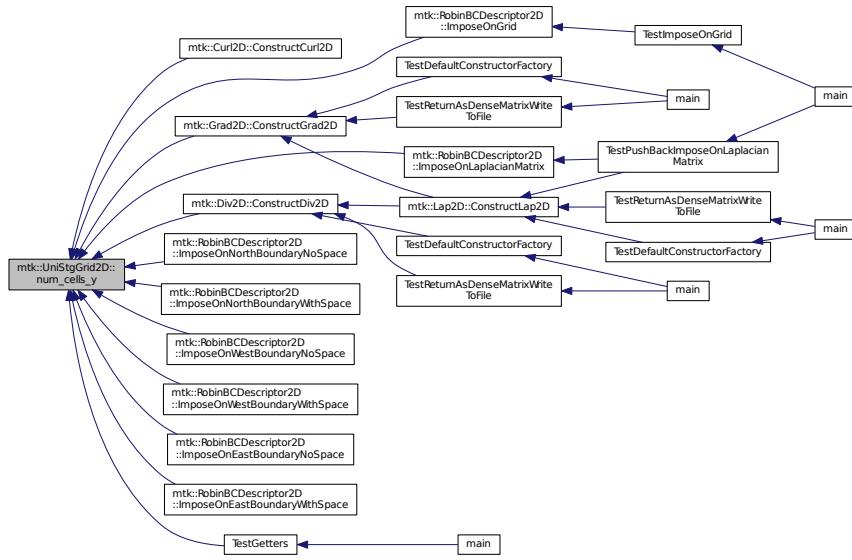
17.25.3.15 int mtk::UniStgGrid2D::num_cells_y() const

Returns

Number of cells of the grid.

Definition at line 246 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.25.3.16 int mtk::UniStgGrid2D::Size () const

Returns

Total number of samples in the grid.

Definition at line 271 of file [mtk_uni_stg_grid_2d.cc](#).

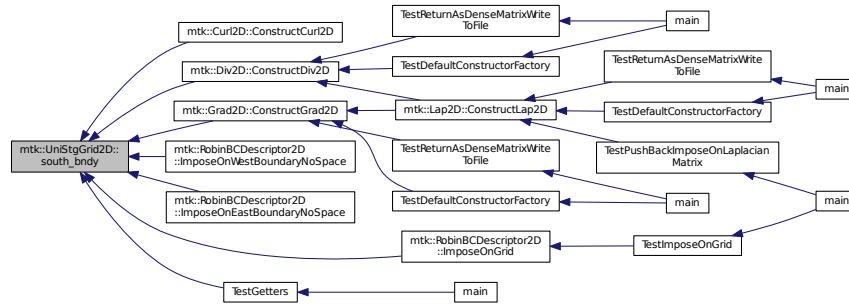
17.25.3.17 mtk::Real mtk::UniStgGrid2D::south_bndy () const

Returns

South boundary spatial coordinate.

Definition at line 236 of file [mtk_uni_stg_grid_2d.cc](#).

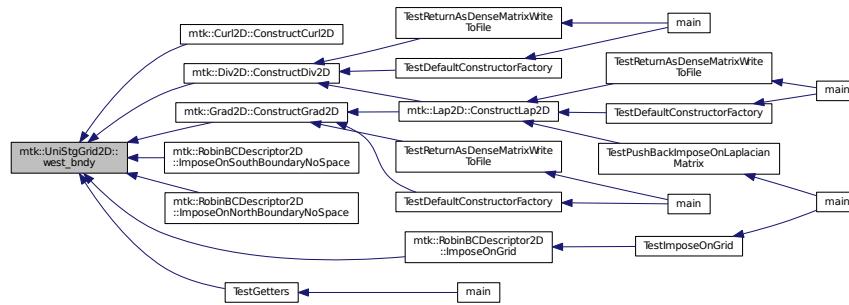
Here is the caller graph for this function:

**17.25.3.18 mtk::Real mtk::UniStgGrid2D::west_bndy () const****Returns**

West boundary spatial coordinate.

Definition at line 211 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:

**17.25.3.19 bool mtk::UniStgGrid2D::WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string field_name) const**

Parameters

in	<i>filename</i>	Name of the output file.
in	<i>space_name_x</i>	Name for the first column of the (spatial) data.
in	<i>space_name_y</i>	Name for the second column of the (spatial) data.
in	<i>field_name</i>	Name for the second column of the (physical field) data.

Returns

Success of the file writing process.

See also

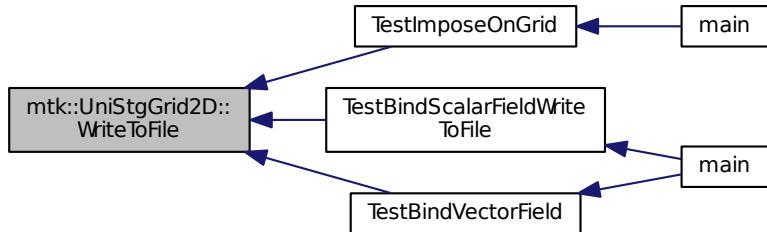
<http://www.gnuplot.info/>

Write the values of the p component, with a null q component.

Write the values of the q component, with a null p component.

Definition at line 438 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.25.4 Friends And Related Function Documentation

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

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

Definition at line 67 of file [mtk_uni_stg_grid_2d.cc](#).

17.25.5 Member Data Documentation

17.25.5.1 `Real mtk::UniStgGrid2D::delta_x_` [private]

Definition at line 309 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 314 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 300 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 301 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 302 of file [mtk_uni_stg_grid_2d.h](#).

17.25.5.6 **Real** `mtk::UniStgGrid2D::east_bndy_` [private]

Definition at line 307 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 304 of file [mtk_uni_stg_grid_2d.h](#).

17.25.5.8 **Real** `mtk::UniStgGrid2D::north_bndy_` [private]

Definition at line 312 of file [mtk_uni_stg_grid_2d.h](#).

17.25.5.9 **int** `mtk::UniStgGrid2D::num_cells_x_` [private]

Definition at line 308 of file [mtk_uni_stg_grid_2d.h](#).

17.25.5.10 **int** `mtk::UniStgGrid2D::num_cells_y_` [private]

Definition at line 313 of file [mtk_uni_stg_grid_2d.h](#).

17.25.5.11 **Real** `mtk::UniStgGrid2D::south_bndy_` [private]

Definition at line 311 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 306 of file [mtk_uni_stg_grid_2d.h](#).

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

- [include/mtk_uni_stg_grid_2d.h](#)

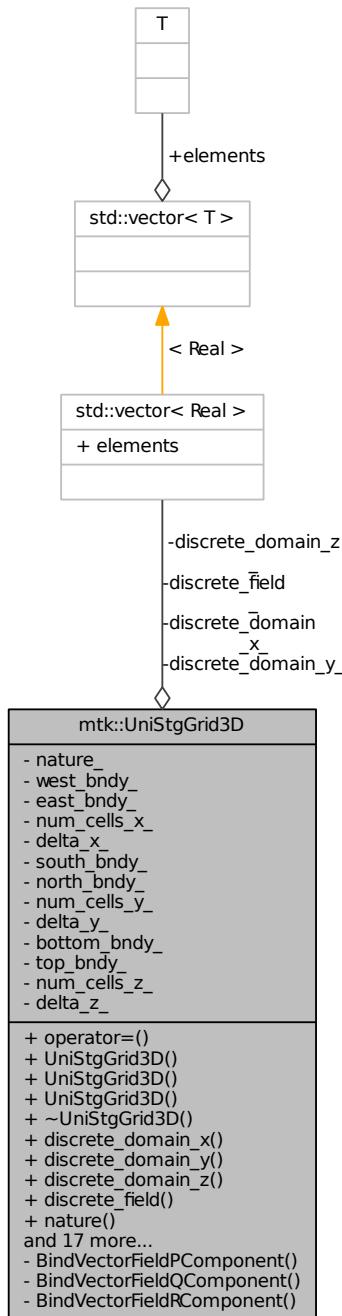
- src/[mtk_uni_stg_grid_2d.cc](#)

17.26 mtk::UniStgGrid3D Class Reference

Uniform 3D Staggered Grid.

```
#include <mtk_uni_stg_grid_3d.h>
```

Collaboration diagram for mtk::UniStgGrid3D:



Public Member Functions

- `UniStgGrid3D operator= (const UniStgGrid3D &in)`

- Overloaded assignment operator.*
- **UniStgGrid3D ()**

Default constructor.
 - **UniStgGrid3D (const UniStgGrid3D &grid)**

Copy constructor.
 - **UniStgGrid3D (const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const Real &south_bndy_y, const Real &north_bndy_y, const int &num_cells_y, const Real &bottom_bndy_z, const Real &top_bndy_z, const int &num_cells_z, const mtk::FieldNature &nature=mtk::FieldNature::SCALAR)**

Construct a grid based on spatial discretization parameters.
 - **~UniStgGrid3D ()**

Destructor.
 - **const Real * discrete_domain_x () const**

Provides access to the grid spatial data.
 - **const Real * discrete_domain_y () const**

Provides access to the grid spatial data.
 - **const Real * discrete_domain_z () const**

Provides access to the grid spatial data.
 - **Real * discrete_field ()**

Provides access to the grid field data.
 - **FieldNature nature () const**

Physical nature of the data bound to the grid.
 - **Real west_bndy () const**

Provides access to west boundary spatial coordinate.
 - **Real east_bndy () const**

Provides access to east boundary spatial coordinate.
 - **int num_cells_x () const**

Provides access to the number of cells of the grid.
 - **Real delta_x () const**

Provides access to the computed \$ x \$.
 - **Real south_bndy () const**

Provides access to south boundary spatial coordinate.
 - **Real north_bndy () const**

Provides access to north boundary spatial coordinate.
 - **int num_cells_y () const**

Provides access to the number of cells of the grid.
 - **Real delta_y () const**

Provides access to the computed \$ y \$.
 - **Real bottom_bndy () const**

Provides access to bottom boundary spatial coordinate.
 - **Real top_bndy () const**

Provides access to top boundary spatial coordinate.
 - **int num_cells_z () const**

Provides access to the number of cells of the grid.
 - **Real delta_z () const**

Provides access to the computed \$ z \$.
 - **bool Bound () const**

Have any field been bound to the grid?

- int `Size () const`
Total number of samples in the grid.
- void `BindScalarField (Real(*ScalarField)(const Real &xx, const Real &yy, const Real &zz))`
Binds a given scalar field to the grid.
- void `BindVectorField (Real(*VectorFieldPComponent)(const Real &xx, const Real &yy, const Real &zz), Real(*VectorFieldQComponent)(const Real &xx, const Real &yy, const Real &zz), Real(*VectorFieldRComponent)(const Real &xx, const Real &yy, const Real &zz))`
Binds a given vector field to the grid.
- bool `WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string space_name_z, std::string field_name) const`
Writes grid to a file compatible with Gnuplot 4.6.

Private Member Functions

- void `BindVectorFieldPComponent (Real(*VectorFieldPComponent)(const Real &xx, const Real &yy, const Real &zz))`
Binds a given component of a vector field to the grid.
- void `BindVectorFieldQComponent (Real(*VectorFieldQComponent)(const Real &xx, const Real &yy, const Real &zz))`
Binds a given component of a vector field to the grid.
- void `BindVectorFieldRComponent (Real(*VectorFieldRComponent)(const Real &xx, const Real &yy, const Real &zz))`
Binds a given component of a vector field to the grid.

Private Attributes

- `std::vector< Real > discrete_domain_x_`
Array of spatial data.
- `std::vector< Real > discrete_domain_y_`
Array of spatial data.
- `std::vector< Real > discrete_domain_z_`
Array of spatial data.
- `std::vector< Real > discrete_field_`
Array of field's data.
- `FieldNature nature_`
Nature of the discrete field.
- `Real west_bndy_`
West boundary spatial coordinate.
- `Real east_bndy_`
East boundary spatial coordinate.
- `int num_cells_x_`
Number of cells discretizing the domain.
- `Real delta_x_`
Computed Δx .
- `Real south_bndy_`
West boundary spatial coordinate.
- `Real north_bndy_`

- int `num_cells_y_`
Number of cells discretizing the domain.
- Real `delta_y_`
Computed Δy .
- Real `bottom_bndy_`
Bottom boundary spatial coordinate.
- Real `top_bndy_`
Top boundary spatial coordinate.
- int `num_cells_z_`
Number of cells discretizing the domain.
- Real `delta_z_`
Computed Δz .

Friends

- std::ostream & `operator<<` (std::ostream &stream, `UniStgGrid3D &in`)
Prints the grid as a tuple of arrays.

17.26.1 Detailed Description

Uniform 3D Staggered Grid.

Definition at line 80 of file [mtk_uni_stg_grid_3d.h](#).

17.26.2 Constructor & Destructor Documentation

17.26.2.1 `mtk::UniStgGrid3D::UniStgGrid3D()`

Definition at line 123 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.2.2 `mtk::UniStgGrid3D::UniStgGrid3D(const UniStgGrid3D & grid)`

Parameters

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

Definition at line 142 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.2.3 `mtk::UniStgGrid3D::UniStgGrid3D(const Real & west_bndy_x, const Real & east_bndy_x, const int & num_cells_x, const Real & south_bndy_y, const Real & north_bndy_y, const int & num_cells_y, const Real & bottom_bndy_z, const Real & top_bndy_z, const int & num_cells_z, const mtk::FieldNature & nature = mtk::FieldNature::SCALAR)`

Parameters

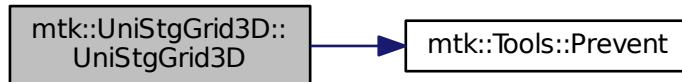
in	<i>west_bndy_x</i>	Coordinate for the west boundary.
in	<i>east_bndy_x</i>	Coordinate for the east boundary.
in	<i>num_cells_x</i>	Number of cells of the required grid.
in	<i>south_bndy_y</i>	Coordinate for the west boundary.
in	<i>north_bndy_y</i>	Coordinate for the east boundary.
in	<i>num_cells_y</i>	Number of cells of the required grid.
in	<i>bottom_bndy_z</i>	Coordinate for the bottom boundary.
in	<i>top_bndy_z</i>	Coordinate for the top boundary.
in	<i>num_cells_z</i>	Number of cells of the required grid.
in	<i>nature</i>	Nature of the discrete field to hold.

See also

[mtk::FieldNature](#)

Definition at line 174 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



17.26.2.4 mtk::UniStgGrid3D::~UniStgGrid3D ()

Definition at line 221 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3 Member Function Documentation

17.26.3.1 void mtk::UniStgGrid3D::BindScalarField (Real(*)(const Real &x, const Real &y, const Real &z) *ScalarField*)

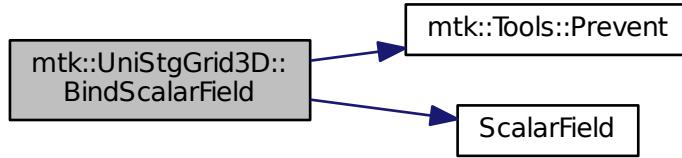
Parameters

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

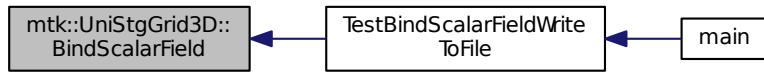
1. Create collection of spatial coordinates for *x*.
2. Create collection of spatial coordinates for *y*.
3. Create collection of spatial coordinates for *z*.
4. Create collection of field samples.

Definition at line 318 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**17.26.3.2 void mtk::UniStgGrid3D::BindVectorField (Real(*)(const Real &xx, const Real &yy, const Real &zz)
VectorFieldPComponent, Real(*)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldQComponent*,
Real(*)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldRComponent*)**

We assume the field to be of the form:

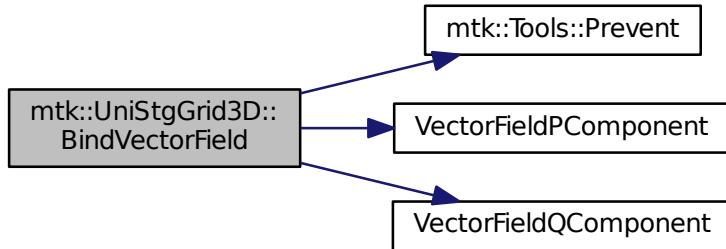
$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

Parameters

in	<i>VectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
in	<i>VectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
in	<i>VectorFieldRComponent</i>	Pointer to the function implementing the \$ r \$ component of the vector field.

Definition at line 415 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



17.26.3.3 void mtk::UniStgGrid3D::BindVectorFieldPComponent (Real(*)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldPComponent*) [private]

We assume the field to be of the form:

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

Parameters

in	<i>BindVectorFieldPComponent</i>	Pointer to the function implementing the \$ p \$ component of the vector field.
----	----------------------------------	---

Definition at line 394 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.4 void mtk::UniStgGrid3D::BindVectorFieldQComponent (Real(*)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldQComponent*) [private]

We assume the field to be of the form:

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

Parameters

in	<i>BindVectorFieldQComponent</i>	Pointer to the function implementing the \$ q \$ component of the vector field.
----	----------------------------------	---

Definition at line 401 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.5 void mtk::UniStgGrid3D::BindVectorFieldRComponent (Real(*)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldRComponent*) [private]

We assume the field to be of the form:

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

Parameters

in	<i>BindVectorField</i> <i>RComponent</i>	Pointer to the function implementing the \$ r \$ component of the vector field.
----	---	---

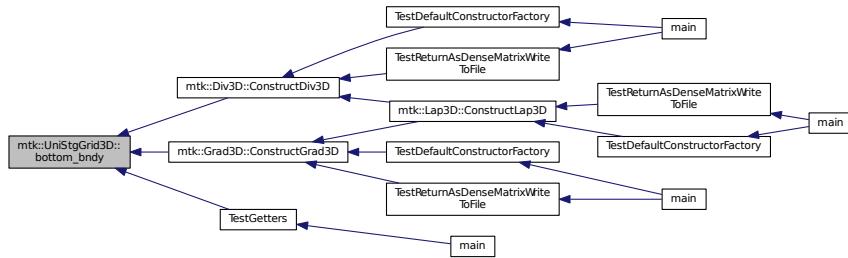
Definition at line 408 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.6 mtk::Real mtk::UniStgGrid3D::bottom_bndy () const**Returns**

Bottom boundary spatial coordinate.

Definition at line 278 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:

**17.26.3.7 bool mtk::UniStgGrid3D::Bound () const****Returns**

True is a field has been bound.

Definition at line 308 of file [mtk_uni_stg_grid_3d.cc](#).

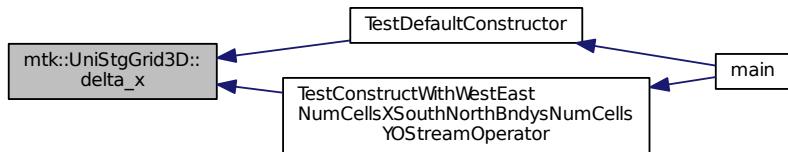
17.26.3.8 mtk::Real mtk::UniStgGrid3D::delta_x () const

Returns

Computed \$ x \$.

Definition at line 243 of file [mtk_uni_stg_grid_3d.cc](#).

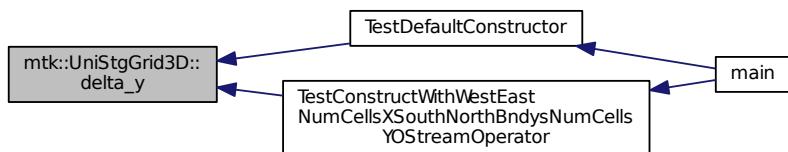
Here is the caller graph for this function:

**17.26.3.9 mtk::Real mtk::UniStgGrid3D::delta_y () const****Returns**

Computed \$ y \$.

Definition at line 268 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:

**17.26.3.10 mtk::Real mtk::UniStgGrid3D::delta_z () const**

Returns

Computed $\$ z \$$.

Definition at line 293 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.26.3.11 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_x() const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 248 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.12 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_y() const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 273 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.13 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_z() const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 298 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.14 `mtk::Real * mtk::UniStgGrid3D::discrete_field()`

Returns

Pointer to the field data.

Definition at line 303 of file [mtk_uni_stg_grid_3d.cc](#).

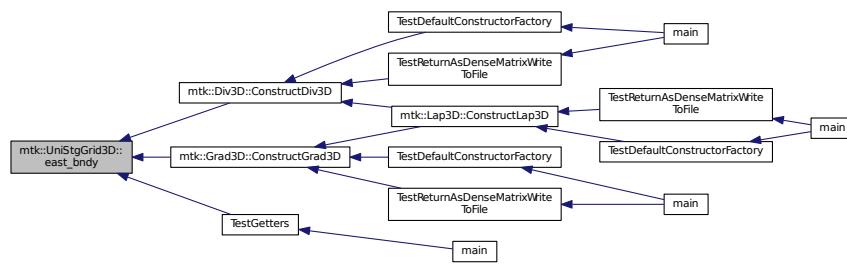
17.26.3.15 mtk::Real mtk::UniStgGrid3D::east_bndy() const

Returns

East boundary spatial coordinate.

Definition at line 233 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.26.3.16 mtk::FieldNature mtk::UniStgGrid3D::nature() const

Returns

Value of an enumeration.

See also

[mtk::FieldNature](#)

Definition at line 223 of file [mtk_uni_stg_grid_3d.cc](#).

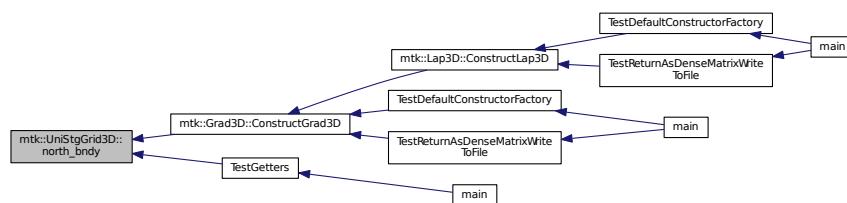
17.26.3.17 mtk::Real mtk::UniStgGrid3D::north_bndy() const

Returns

North boundary spatial coordinate.

Definition at line 258 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



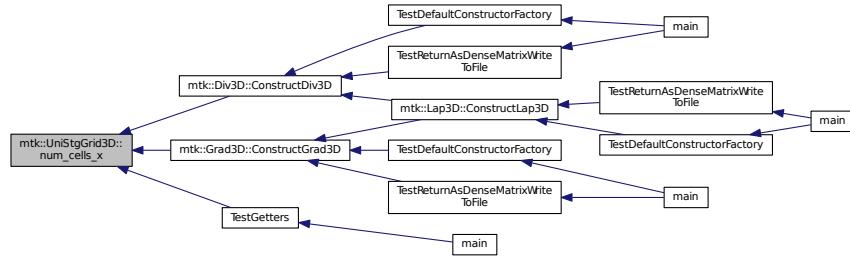
17.26.3.18 int mtk::UniStgGrid3D::num_cells_x() const

Returns

Number of cells of the grid.

Definition at line 238 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



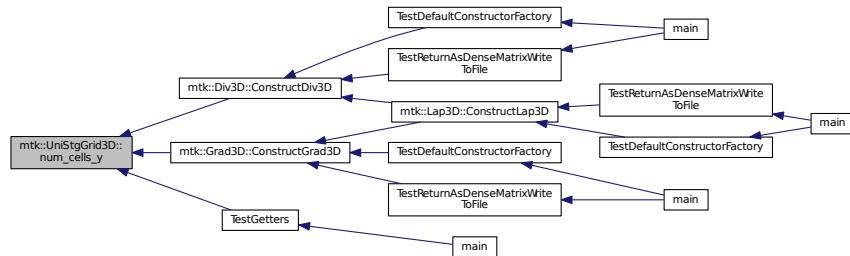
17.26.3.19 int mtk::UniStgGrid3D::num_cells_y() const

Returns

Number of cells of the grid.

Definition at line 263 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



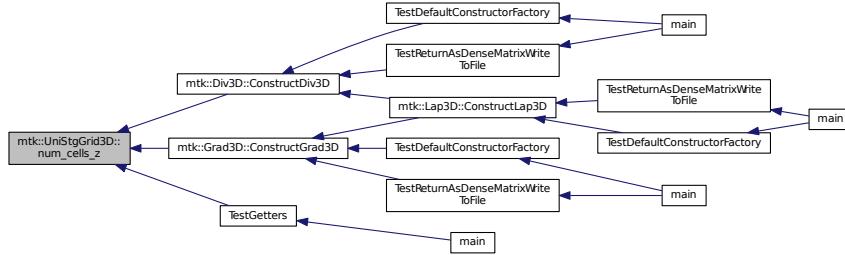
17.26.3.20 int mtk::UniStgGrid3D::num_cells_z() const

Returns

Number of cells of the grid.

Definition at line 288 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.26.3.21 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator= (const UniStgGrid3D & in)

Parameters

in	in	Given grid.
----	----	-------------

Returns

Copy of the given grid.

Definition at line 116 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.3.22 int mtk::UniStgGrid3D::Size () const

Returns

Total number of samples in the grid.

Definition at line 313 of file [mtk_uni_stg_grid_3d.cc](#).

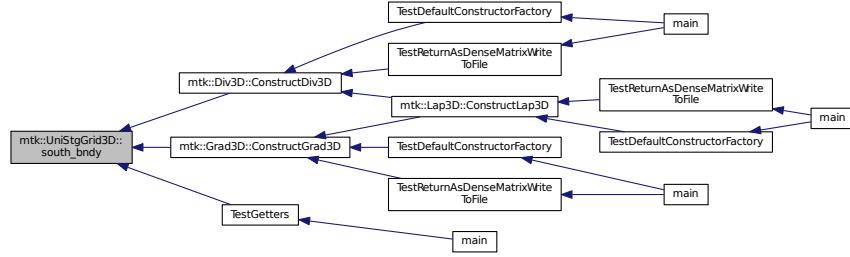
17.26.3.23 mtk::Real mtk::UniStgGrid3D::south_bndy () const

Returns

South boundary spatial coordinate.

Definition at line 253 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



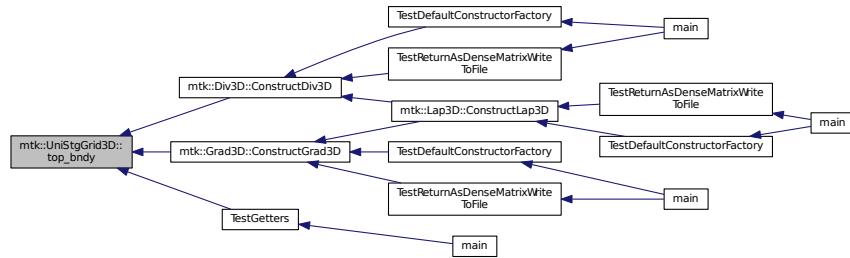
17.26.3.24 mtk::Real mtk::UniStgGrid3D::top_bndy() const

Returns

Top boundary spatial coordinate.

Definition at line 283 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



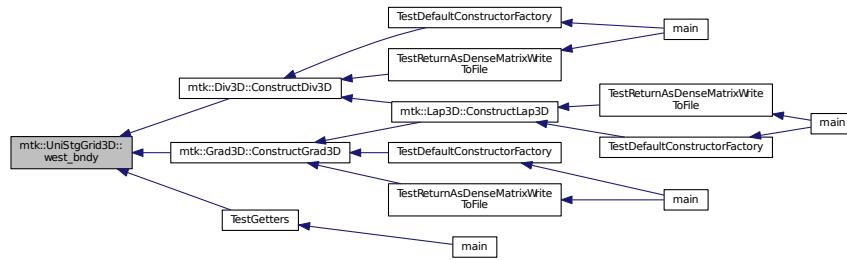
17.26.3.25 mtk::Real mtk::UniStgGrid3D::west_bndy() const

Returns

West boundary spatial coordinate.

Definition at line 228 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.26.3.26 `bool mtk::UniStgGrid3D::WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string space_name_z, std::string field_name) const`

Parameters

in	<i>filename</i>	Name of the output file.
in	<i>space_name_x</i>	Name for the first column of the (spatial) data.
in	<i>space_name_y</i>	Name for the second column of the (spatial) data.
in	<i>space_name_z</i>	Name for the third column of the (spatial) data.
in	<i>field_name</i>	Name for the second column of the (physical field) data.

Returns

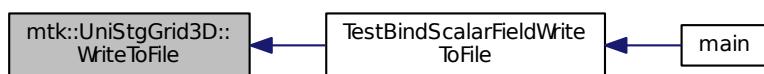
Success of the file writing process.

See also

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

Definition at line 435 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.26.4 Friends And Related Function Documentation

17.26.4.1 `std::ostream& operator<< (std::ostream & stream, mtk::UniStgGrid3D & in)` [friend]

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

Definition at line 67 of file [mtk_uni_stg_grid_3d.cc](#).

17.26.5 Member Data Documentation

17.26.5.1 `Real mtk::UniStgGrid3D::bottom_bndy_` [private]

Definition at line 403 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.2 `Real mtk::UniStgGrid3D::delta_x_` [private]

Definition at line 396 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.3 `Real mtk::UniStgGrid3D::delta_y_` [private]

Definition at line 401 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.4 `Real mtk::UniStgGrid3D::delta_z_` [private]

Definition at line 406 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.5 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_x_` [private]

Definition at line 386 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.6 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_y_` [private]

Definition at line 387 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.7 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_z_` [private]

Definition at line 388 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.8 `std::vector<Real> mtk::UniStgGrid3D::discrete_field_` [private]

Definition at line 389 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.9 `Real mtk::UniStgGrid3D::east_bndy_` [private]

Definition at line 394 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.10 **FieldNature** `mtk::UniStgGrid3D::nature_` [private]

Definition at line 391 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.11 **Real** `mtk::UniStgGrid3D::north_bndy_` [private]

Definition at line 399 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.12 **int** `mtk::UniStgGrid3D::num_cells_x_` [private]

Definition at line 395 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.13 **int** `mtk::UniStgGrid3D::num_cells_y_` [private]

Definition at line 400 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.14 **int** `mtk::UniStgGrid3D::num_cells_z_` [private]

Definition at line 405 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.15 **Real** `mtk::UniStgGrid3D::south_bndy_` [private]

Definition at line 398 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.16 **Real** `mtk::UniStgGrid3D::top_bndy_` [private]

Definition at line 404 of file [mtk_uni_stg_grid_3d.h](#).

17.26.5.17 **Real** `mtk::UniStgGrid3D::west_bndy_` [private]

Definition at line 393 of file [mtk_uni_stg_grid_3d.h](#).

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

- `include/mtk_uni_stg_grid_3d.h`
- `src/mtk_uni_stg_grid_3d.cc`

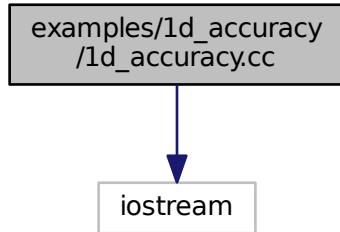
Chapter 18

File Documentation

18.1 examples/1d_accuracy/1d_accuracy.cc File Reference

Check the accuracy of mimetic operators.

```
#include <iostream>
Include dependency graph for 1d_accuracy.cc:
```



Functions

- int `main ()`

18.1.1 Detailed Description

Author

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

Definition in file [1d_accuracy.cc](#).

18.1.2 Function Documentation

18.1.2.1 int main()

Definition at line 321 of file [1d_accuracy.cc](#).

18.2 1d_accuracy.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.cs.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.cs.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <cmath>
00059
00060 #include <iostream>
00061 #include <fstream>
00062
00063 #include <array>
00064 #include <vector>
00065
00066 #include "mtk.h"
00067
00070
00071 mtk::Real Polynomial(const mtk::Real &xx, const std::vector<mtk::Real> &pp) {
00072
00073     mtk::Real sum{};
00074
00075     mtk::Real kk{pp[0]};
00076
00077     for (int ii = 0; ii <= kk; ++ii) {

```

```

00078     sum += pow(xx, ii);
00079 }
00080 return sum;
00081 }
00082
00083 mtk::Real PolynomialDerivative1(const mtk::Real &xx,
00084                                     const std::vector<mtk::Real> &pp) {
00085
00086     mtk::Real sum{};
00087
00088     mtk::Real kk{pp[0]};
00089
00090     for (int ii = 1; ii <= kk; ++ii) {
00091         sum += ii * pow(xx, ii - 1);
00092     }
00093     return sum;
00094 }
00095
00096 mtk::Real HomogeneousDirichlet(const mtk::Real &tt,
00097                                   const std::vector<mtk::Real> &pp) {
00098
00099     return mtk::kOne;
00100 }
00101
00102 int main () {
00103
00104     std::cout << "Example: Checking the accuracy of the mimetic operators." <<
00105         std::endl;
00106
00107     const int max_order{14};
00108
00109     mtk::Real aa = 0.0;
00110     mtk::Real bb = 1.0;
00111
00112     std::array<mtk::Real, 3> epsilons{1.0e-3, 1.0e-6, 1.0e-9};
00113
00114     std::ofstream output_dat_file; // Output file.
00115
00116
00117     output_dat_file.open("accuracy_grad.tex");
00118
00119
00120     if (!output_dat_file.is_open()) {
00121         std::cerr << "Could not open data file." << std::endl;
00122         return EXIT_FAILURE;
00123     }
00124
00125     output_dat_file << "\\begin{tabular}[c]{c:ccc}" << std::endl;
00126     output_dat_file << "\\toprule" << std::endl;
00127     output_dat_file << "$k$ & \\multicolumn{3}{c}{Relative error} \\"\\\"\\"
00128         << std::endl;
00129     output_dat_file << " & $\\epsilon = 1\\times 10^{-3}$ &
00130         "$\\epsilon = 1\\times 10^{-6}$ &
00131         "$\\epsilon = 1\\times 10^{-9}$ \\"\\\"\\"
00132     output_dat_file << "\\midrule" << std::endl;
00133
00134     mtk::Grad1D grad;
00135
00136     for (int order = 2; order <= max_order; order += 2) {
00137
00138         output_dat_file << order;
00139
00140         for (mtk::Real &epsilon: epsilons) {
00141
00142             if (!grad.ConstructGrad1D(order, epsilon)) {
00143                 std::cerr << "Mimetic gradient could not be built." << std::endl;
00144                 return EXIT_FAILURE;
00145             }
00146
00147             int nn{500};
00148
00149             mtk::UniStgGrid1D gg(aa, bb, nn);
00150
00151             std::vector<mtk::Real> order_in;
00152
00153             order_in.push_back((mtk::Real) order);
00154
00155             gg.BindScalarField(Polynomial, order_in);
00156
00157             if (!gg.WriteToFile("1d_accuracy_gg.dat", "x", "p(x)")) {
00158                 std::cerr << "Polynomial could not be written to file." << std::endl;
00159                 return EXIT_FAILURE;

```

```

00160      }
00161
00162     mtk::DenseMatrix gradm{grad.ReturnAsDenseMatrix(gg)};
00163
00164     if (!gradm.WriteToFile("1d_accuracy_gradm.dat")) {
00165         std::cerr << "Grad matrix could not be written to disk." << std::endl;
00166         return EXIT_FAILURE;
00167     }
00168
00169     mtk::UniStgGrid1D out(gg.west_bndy_x(),
00170                           gg.east_bndy_x(),
00171                           gg.num_cells_x(),
00172                           mtk::FieldNature::VECTOR);
00173
00174     out.GenerateDiscreteDomainX();
00175     out.ReserveDiscreteField();
00176
00177     mtk::OperatorApplicator::ApplyDenseMatrixGradientOn1DGrid
00178     (gradm, gg, out);
00179
00180     if (!out.WriteToFile("1d_accuracy_out.dat", "x", "~p'(x)")) {
00181         std::cerr << "Comp Grad Pol could not be written to file." << std::endl;
00182         return EXIT_FAILURE;
00183     }
00184
00185     mtk::UniStgGrid1D gg_prime(aa, bb, nn,
00186                               mtk::FieldNature::VECTOR);
00187
00188     gg_prime.BindVectorField(PolynomialDerivative1, order_in);
00189
00190     if (!gg_prime.WriteToFile("1d_accuracy_gg_prime.dat", "x", "p'(x)")) {
00191         std::cerr << "Grad Poly could not be written to file." << std::endl;
00192         return EXIT_FAILURE;
00193     }
00194
00195     mtk::Real relative_norm_2_error{};
00196
00197     relative_norm_2_error =
00198         mtk::BLASAdapter::RelNorm2Error(out.discrete_field(),
00199                                         gg_prime.discrete_field(),
00200                                         gg_prime.num_cells_x());
00201
00202     output_dat_file << " & " << relative_norm_2_error;
00203
00204     output_dat_file << "\\\\" << std::endl;
00205
00206     output_dat_file << "\\bottomrule" << std::endl;
00207     output_dat_file << "\\end{tabular}" << std::endl;
00208
00209     output_dat_file.close();
00210
00211
00212     output_dat_file.open("accuracy_div.tex");
00213
00214     if (!output_dat_file.is_open()) {
00215         std::cerr << "Could not open data file." << std::endl;
00216         return EXIT_FAILURE;
00217     }
00218
00219     output_dat_file << "\\begin{tabular}[c]{c:ccc}" << std::endl;
00220     output_dat_file << "\\toprule";
00221     output_dat_file << "$k\$ & \\multicolumn{3}{c}{Relative error} \\\\\" <<
00222     std::endl;
00223     output_dat_file << " & \$\\epsilon = 1\\times 10^{-3}\$ & "
00224     " \$\\epsilon = 1\\times 10^{-6}\$ & "
00225     " \$\\epsilon = 1\\times 10^{-9}\$ \\\\\" << std::endl;
00226     output_dat_file << "\\midrule" << std::endl;
00227
00228     mtk::Div1D div;
00229
00230     for (int order = 2; order <= max_order; order += 2) {
00231
00232         output_dat_file << order;
00233
00234         for (mtk::Real &epsilon: epsilons) {
00235
00236             if (!div.ConstructDiv1D(order, epsilon)) {
00237                 std::cerr << "Mimetic divergence could not be built." << std::endl;
00238                 return EXIT_FAILURE;
00239             }
00240         }
00241     }
00242
00243     output_dat_file << "\\end{tabular}" << std::endl;
00244
00245     output_dat_file << "\\hline" << std::endl;
00246
00247     output_dat_file << "\\bottomrule" << std::endl;
00248
00249     output_dat_file << "\\end{table}" << std::endl;
00250
00251     output_dat_file << "\\end{document}" << std::endl;
00252
00253     output_dat_file.close();
00254
00255     std::cout << "Accuracy data generated successfully." << std::endl;
00256
00257     return 0;
00258 }
```

```

00240      }
00241
00242     int nn{500};
00243
00244     mtk::UniStgGrid1D vv(aa, bb, nn, mtk::FieldNature::VECTOR);
00245
00246     std::vector<mtk::Real> order_in,
00247
00248     order_in.push_back((mtk::Real) order);
00249
00250     vv.BindVectorField(Polynomial, order_in);
00251
00252     if (!vv.WriteToFile("1d_accuracy_vv.dat", "x", "||p(x)||")) {
00253         std::cerr << "Polynomial could not be written to file." << std::endl;
00254         return EXIT_FAILURE;
00255     }
00256
00257     mtk::DenseMatrix divm{div.ReturnAsDenseMatrix(vv)};
00258
00259     mtk::RobinBCDescriptor1D robin_bc_desc_1d;
00260
00261     robin_bc_desc_1d.PushBackWestCoeff(HomogeneousDirichlet);
00262     robin_bc_desc_1d.PushBackEastCoeff(HomogeneousDirichlet);
00263
00264     if (!robin_bc_desc_1d.ImposeOnDivergenceMatrix(div, divm)) {
00265         std::cerr << "BCs could not be bound to the matrix." << std::endl;
00266         return EXIT_FAILURE;
00267     }
00268
00269     if (!divm.WriteToFile("1d_accuracy_divm.dat")) {
00270         std::cerr << "Div matrix could not be written to disk." << std::endl;
00271         return EXIT_FAILURE;
00272     }
00273
00274     mtk::UniStgGrid1D out2(vv.west_bndy_x(),
00275                           vv.east_bndy_x(),
00276                           vv.num_cells_x());
00277
00278     out2.GenerateDiscreteDomainX();
00279     out2.ReserveDiscreteField();
00280
00281     mtk::OperatorApplicator::ApplyDenseMatrixDivergenceOn1DGrid
00282     (divm,
00283
00284
00285     vv,
00286     out2);
00287
00288     if (!out2.WriteToFile("1d_accuracy_out2.dat", "x", "~p'(x)")) {
00289         std::cerr << "Comp Grad Pol could not be written to file." << std::endl;
00290         return EXIT_FAILURE;
00291     }
00292
00293     mtk::UniStgGrid1D vv_prime(aa, bb, nn);
00294
00295     vv_prime.BindScalarField(PolynomialDerivative1, order_in);
00296
00297     if (!vv_prime.WriteToFile("1d_accuracy_vv_prime.dat", "x", "p'(x)")) {
00298         std::cerr << "Div Poly could not be written to file." << std::endl;
00299         return EXIT_FAILURE;
00300     }
00301
00302     mtk::Real relative_norm_2_error{};
00303
00304     relative_norm_2_error =
00305         mtk::BLASAdapter::RelNorm2Error(out2.discrete_field(),
00306                                         vv_prime.discrete_field(),
00307                                         vv_prime.num_cells_x());
00308
00308     output_dat_file << " & " << relative_norm_2_error;
00309
00310     output_dat_file << "\\\\" << std::endl;
00311
00312     output_dat_file << "\\bottomrule" << std::endl;
00313     output_dat_file << "\\end{tabular}" << std::endl;
00314
00315     output_dat_file.close();
00316 }
00317 #else
00318 #include <iostream>
00319 using std::cout;

```

```

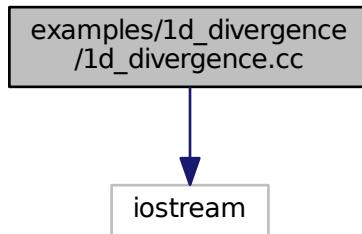
00320 using std::endl;
00321 int main () {
00322     cout << "This code HAS to be compiled with support for C++11." << endl;
00323     cout << "Exiting..." << endl;
00324     return EXIT_SUCCESS;
00325 }
00326 #endif

```

18.3 examples/1d_divergence/1d_divergence.cc File Reference

Creates instances of a 1D divergence as computed by the CBS algorithm.

```
#include <iostream>
Include dependency graph for 1d_divergence.cc:
```



Functions

- **int main ()**

18.3.1 Detailed Description

Author

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

Definition in file [1d_divergence.cc](#).

18.3.2 Function Documentation

18.3.2.1 int main ()

Definition at line [102](#) of file [1d_divergence.cc](#).

18.4 1d_divergence.cc

00001

```
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Instances of a 1D divergence as computed by the CBS "
00067         "algorithm." << std::endl;
00068
00069
00070     std::ofstream output_tex_file;
00071
00072     int max_order{14};
00073
00074     for (int order = 2; order <= max_order; order += 2) {
00075
00076         std::string output_tex_file_name("div_1d_" + std::to_string(order) +
00077             ".tex");
00078
00079         output_tex_file.open(output_tex_file_name);
00080
00081         mtk::Div1D div;
00082
00083         bool assertion = div.ConstructDiv1D(order);
00084         if (!assertion) {
00085             std::cerr << "Mimetic div (order" + std::to_string(order) +
00086                 ") could not be built." << std::endl;
00087             return EXIT_FAILURE;
00088         }
00089     }
```

```

00090     output_tex_file << "\\begin{verbatim}" << std::endl;
00091     output_tex_file << div << std::endl;
00092     output_tex_file << "\\end{verbatim}" << std::endl;
00093     output_tex_file.close();
00094 }
00095 }
00096 }
00097
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   return EXIT_SUCCESS;
00106 }
00107 #endif

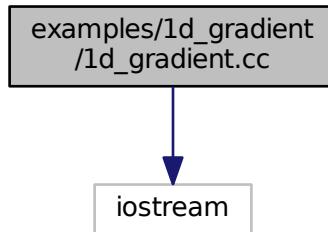
```

18.5 examples/1d_gradient/1d_gradient.cc File Reference

Creates instances of a 1D gradient as computed by the CBS algorithm.

#include <iostream>

Include dependency graph for 1d_gradient.cc:



Functions

- int **main ()**

18.5.1 Detailed Description

Author

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

Definition in file [1d_gradient.cc](#).

18.5.2 Function Documentation

18.5.2.1 int main()

Definition at line 102 of file [1d_gradient.cc](#).

18.6 1d_gradient.cc

```
00001
00008  /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Instances of a 1D gradient as computed by the CBS "
00067     "algorithm." << std::endl;
00068
00069
00070     std::ofstream output_tex_file;
00071
00072     int max_order{14};
00073
00074     for (int order = 2; order <= max_order; order += 2) {
00075
00076         std::string output_tex_file_name("grad_1d_" + std::to_string(order) +
```

```

00078     ".tex");
00079
00080     output_tex_file.open(output_tex_file_name);
00081
00082     mtk::Grad1D grad;
00083
00084     bool assertion = grad.ConstructGrad1D(order);
00085     if (!assertion) {
00086         std::cerr << "Mimetic grad (order" + std::to_string(order) +
00087             ") could not be built." << std::endl;
00088         return EXIT_FAILURE;
00089     }
00090
00091     output_tex_file << "\\begin{verbatim}" << std::endl;
00092     output_tex_file << grad << std::endl;
00093     output_tex_file << "\\end{verbatim}" << std::endl;
00094     output_tex_file.close();
00095 }
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     return EXIT_SUCCESS;
00106 }
00107 #endif

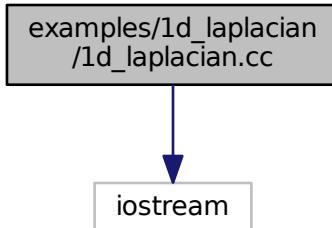
```

18.7 examples/1d_laplacian/1d_laplacian.cc File Reference

Creates instances of a 1D Laplacian as computed by the CBS algorithm.

#include <iostream>

Include dependency graph for 1d_laplacian.cc:



Functions

- int **main** ()

18.7.1 Detailed Description

Author

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

Definition in file [1d_laplacian.cc](#).

18.7.2 Function Documentation

18.7.2.1 int main()

Definition at line 102 of file [1d_laplacian.cc](#).

18.8 1d_laplacian.cc

```
00001
00002  /*
00003 Copyright (C) 2016, Computational Science Research Center, San Diego State
00004 University. All rights reserved.
00005
00006 Redistribution and use in source and binary forms, with or without modification,
00007 are permitted provided that the following conditions are met:
00008
00009 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00010 and a copy of the modified files should be reported once modifications are
00011 completed, unless these modifications are made through the project's GitHub
00012 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00013 should be developed and included in any deliverable.
00014
00015 2. Redistributions of source code must be done through direct
00016 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00017
00018 3. Redistributions in binary form must reproduce the above copyright notice,
00019 this list of conditions and the following disclaimer in the documentation and/or
00020 other materials provided with the distribution.
00021
00022 4. Usage of the binary form on proprietary applications shall require explicit
00023 prior written permission from the the copyright holders, and due credit should
00024 be given to the copyright holders.
00025
00026 5. Neither the name of the copyright holder nor the names of its contributors
00027 may be used to endorse or promote products derived from this software without
00028 specific prior written permission.
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
```

```

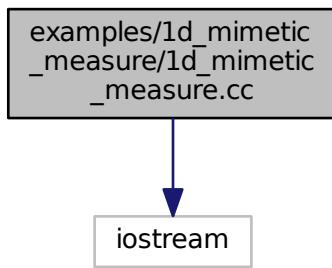
00064 int main () {
00065     std::cout << "Example: Instances of a 1D Laplacian as computed by the CBS "
00066         "algorithm." << std::endl;
00067
00068
00069     std::ofstream output_tex_file;
00070
00071     int max_order{14};
00072
00073     for (int order = 2; order <= max_order; order += 2) {
00074
00075         std::string output_tex_file_name("lap_1d_" + std::to_string(order) +
00076             ".tex");
00077
00078         output_tex_file.open(output_tex_file_name);
00079
00080         mtk::Lap1D lap;
00081
00082         bool assertion = lap.ConstructLap1D(order);
00083         if (!assertion) {
00084             std::cerr << "Mimetic lap (order" + std::to_string(order) +
00085                 ") could not be built." << std::endl;
00086             return EXIT_FAILURE;
00087         }
00088
00089         output_tex_file << "\\begin{verbatim}" << std::endl;
00090         output_tex_file << lap << std::endl;
00091         output_tex_file << "\\end{verbatim}" << std::endl;
00092         output_tex_file.close();
00093
00094     }
00095 }
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     return EXIT_SUCCESS;
00106 }
00107 #endif

```

18.9 examples/1d_mimetic_measure/1d_mimetic_measure.cc File Reference

Compute the mimetic measure of different mimetic operators.

```
#include <iostream>
Include dependency graph for 1d_mimetic_measure.cc:
```



Functions

- int `main ()`

18.9.1 Detailed Description

Author

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

Definition in file [1d_mimetic_measure.cc](#).

18.9.2 Function Documentation

18.9.2.1 int main ()

Definition at line [120](#) of file [1d_mimetic_measure.cc](#).

18.10 1d_mimetic_measure.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055

```

```

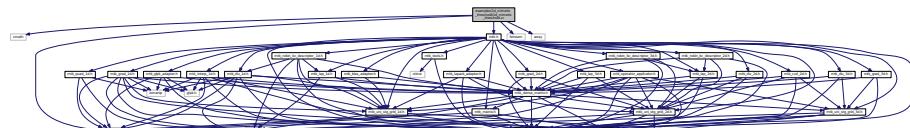
00056 #if __cplusplus == 201103L
00057
00058 #include <cmath>
00059
00060 #include <iostream>
00061 #include <fstream>
00062
00063 #include <array>
00064
00065 #include "mtk.h"
00066
00067 int main () {
00068
00069     std::cout << "Example: Computing the mimetic measure of gradient, "
00070         "divergence, and Laplacian operators." << std::endl;
00071
00072     const int max_order{14};
00073
00074     std::ofstream output_dat_file; // Output file.
00075
00076     output_dat_file.open("mimetic_measure.tex");
00077
00078     if (!output_dat_file.is_open()) {
00079         std::cerr << "Could not open data file." << std::endl;
00080         return EXIT_FAILURE;
00081     }
00082
00083     output_dat_file << "\\begin{tabular}[c]{c:ccc}" << std::endl;
00084     output_dat_file << "\\toprule";
00085     output_dat_file << "$\\mu(\\breve{\\mathbf{G}})^k_x $" &
00086         "$\\mu(\\breve{\\mathbf{D}})^k_x $" &
00087         "$\\mu(\\breve{\\mathbf{L}})^k_x $" \\\"\\\" << std::endl;
00088     output_dat_file << "\\midrule";
00089
00090     mtk::Grad1D grad;
00091     mtk::Div1D div;
00092     mtk::Lap1D lap;
00093
00094     for (int order = 2; order <= max_order; order += 2) {
00095
00096         bool go1{grad.ConstructGrad1D(order)};
00097         bool go2{div.ConstructDiv1D(order)};
00098         bool go3{lap.ConstructLap1D(order)};
00099
00100         if (go1 && go2 && go3) {
00101             output_dat_file << order << " & " << grad.mimetic_measure() << " & " <<
00102                 div.mimetic_measure() << " & " << lap.mimetic_measure() << " \\\"\\\" <<
00103                 std::endl;
00104
00105         } else {
00106             std::cerr << "Mimetic operator could not be built." << std::endl;
00107             return EXIT_FAILURE;
00108         }
00109     }
00110
00111     output_dat_file << "\\bottomrule" << std::endl;
00112     output_dat_file << "\\end{tabular}" << std::endl;
00113
00114     output_dat_file.close();
00115 }
00116 #else
00117 #include <iostream>
00118 using std::cout;
00119 using std::endl;
00120 int main () {
00121     cout << "This code HAS to be compiled with support for C++11." << endl;
00122     cout << "Exiting..." << endl;
00123     return EXIT_SUCCESS;
00124 }
00125 #endif

```

18.11 examples/1d_mimetic_threshold/1d_mimetic_threshold.cc File Reference

Perform a sensitivity analysis of the mimetic threshold.

```
#include <cmath>
#include <iostream>
#include <fstream>
#include <array>
#include "mtk.h"
Include dependency graph for 1d_mimetic_threshold.cc:
```



Functions

- int `main ()`

18.11.1 Detailed Description

Author

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

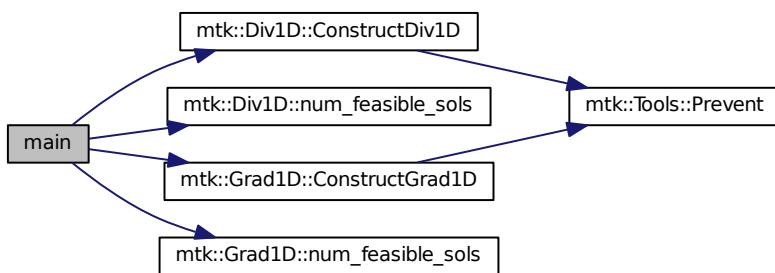
Definition in file [1d_mimetic_threshold.cc](#).

18.11.2 Function Documentation

18.11.2.1 int main ()

Definition at line [65](#) of file [1d_mimetic_threshold.cc](#).

Here is the call graph for this function:



18.12 1d_mimetic_threshold.cc

00001

```

00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <cmath>
00057
00058 #include <iostream>
00059 #include <fstream>
00060
00061 #include <array>
00062
00063 #include "mtk.h"
00064
00065 int main () {
00066
00067     std::cout << "Example: Sensitivity analysis of the mimetic threshold when "
00068         "constructing 1D gradient and divergence operators." << std::endl;
00069
00070     std::array<int, 4> orders = {8, 10, 12, 14};
00071
00072     const int num_samples{10};
00073
00074     std::vector<mtk::Real> thresholds(num_samples);
00075
00076     thresholds.at(0) = 1e0;
00077     for (size_t ii = 1; ii < thresholds.size(); ++ii) {
00078         thresholds.at(ii) = thresholds.at(ii - 1)/10.0;
00079     }
00080
00081     std::array<mtk::Real, num_samples> num_feasible_sols{};
00082
00083     std::ofstream output_dat_file; // Output file.
00084
00085     for (int kk: orders) {
00086
00087         output_dat_file.open("mimetic_threshold_div_" + std::to_string(kk) +
00088             ".dat");
00089
00090         if (!output_dat_file.is_open()) {

```

```

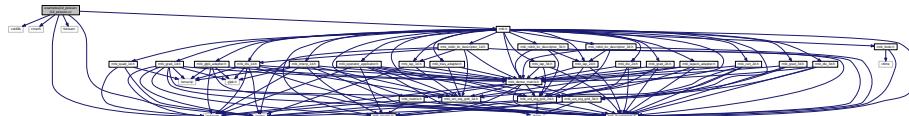
00091     std::cerr << "Could not open data file." << std::endl;
00092     return EXIT_FAILURE;
00093 }
00094
00095 output_dat_file << "# " << 'e' << ' ' << 'n' << std::endl;
00096
00097 for (size_t ii = 0; ii < thresholds.size(); ++ii) {
00098
00099     mtk::Real epsilon{thresholds.at(ii)};
00100
00101     mtk::Div1D div;
00102
00103     if (div.ConstructDiv1D(kk, epsilon)) {
00104         num_feasible_sols[ii] = div.num_feasible_sols();
00105     } else {
00106         std::cerr << "Mimetic divergence could not be built." << std::endl;
00107         return EXIT_FAILURE;
00108     }
00109 }
00110
00111 for (unsigned int ii = 0; ii < thresholds.size(); ++ii) {
00112     output_dat_file << thresholds[ii] << ' ' << num_feasible_sols[ii] <<
00113         std::endl;
00114 }
00115
00116 output_dat_file.close();
00117 }
00118
00119 for (int kk: orders) {
00120
00121     output_dat_file.open("mimetic_threshold_grad_" + std::to_string(kk) +
00122         ".dat");
00123
00124     if (!output_dat_file.is_open()) {
00125         std::cerr << "Could not open data file." << std::endl;
00126         return EXIT_FAILURE;
00127     }
00128
00129     output_dat_file << "# " << 'e' << ' ' << 'n' << std::endl;
00130
00131     for (size_t ii = 0; ii < thresholds.size(); ++ii) {
00132
00133         mtk::Real epsilon{thresholds.at(ii)};
00134
00135         mtk::Grad1D grad;
00136
00137         if (grad.ConstructGrad1D(kk, epsilon)) {
00138             num_feasible_sols[ii] = grad.num_feasible_sols();
00139         } else {
00140             std::cerr << "Mimetic gradient could not be built." << std::endl;
00141             return EXIT_FAILURE;
00142         }
00143     }
00144
00145     for (unsigned int ii = 0; ii < thresholds.size(); ++ii) {
00146         output_dat_file << thresholds[ii] << ' ' << num_feasible_sols[ii] <<
00147             std::endl;
00148     }
00149
00150     output_dat_file.close();
00151 }
00152 }
```

18.13 examples/1d_poisson/1d_poisson.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <vector>
#include "mtk.h"
```

Include dependency graph for 1d_poisson.cc:



Functions

- mtk::Real Alpha (const mtk::Real &tt, const std::vector< mtk::Real > &pp)
- mtk::Real Beta (const mtk::Real &tt, const std::vector< mtk::Real > &pp)
- mtk::Real Omega (const mtk::Real &tt)
- mtk::Real Epsilon (const mtk::Real &tt)
- mtk::Real Source (const mtk::Real &xx, const std::vector< mtk::Real > &pp)
- mtk::Real KnownSolution (const mtk::Real &xx, const std::vector< mtk::Real > &pp)
- int main ()

18.13.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for $x \in \Omega = [a, b] = [0, 1]$.

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where $\lambda = -1$ is a real-valued parameter.

We consider Robin's boundary conditions of the form:

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

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

where $\alpha = -\exp(\lambda)$, $\beta = \lambda^{-1}(\exp(\lambda) - 1.0)$, $\omega = -1$, and $\varepsilon = 0$.

The analytical solution for this problem is given by:

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

The mimetic counterpart of this equation is:

$$-\check{\mathbf{I}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

Definition in file [1d_poisson.cc](#).

18.13.2 Function Documentation

18.13.2.1 `mtk::Real Alpha (const mtk::Real & tt, const std::vector< mtk::Real > & pp)`

Definition at line 99 of file [1d_poisson.cc](#).

Here is the caller graph for this function:



18.13.2.2 `mtk::Real Beta (const mtk::Real & tt, const std::vector< mtk::Real > & pp)`

Definition at line 106 of file [1d_poisson.cc](#).

Here is the caller graph for this function:



18.13.2.3 `mtk::Real Epsilon (const mtk::Real & tt)`

Definition at line 118 of file [1d_poisson.cc](#).

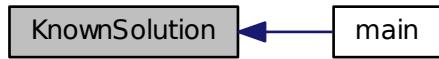
Here is the caller graph for this function:



18.13.2.4 mtk::Real KnownSolution (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 130 of file [1d_poisson.cc](#).

Here is the caller graph for this function:

**18.13.2.5 int main ()**

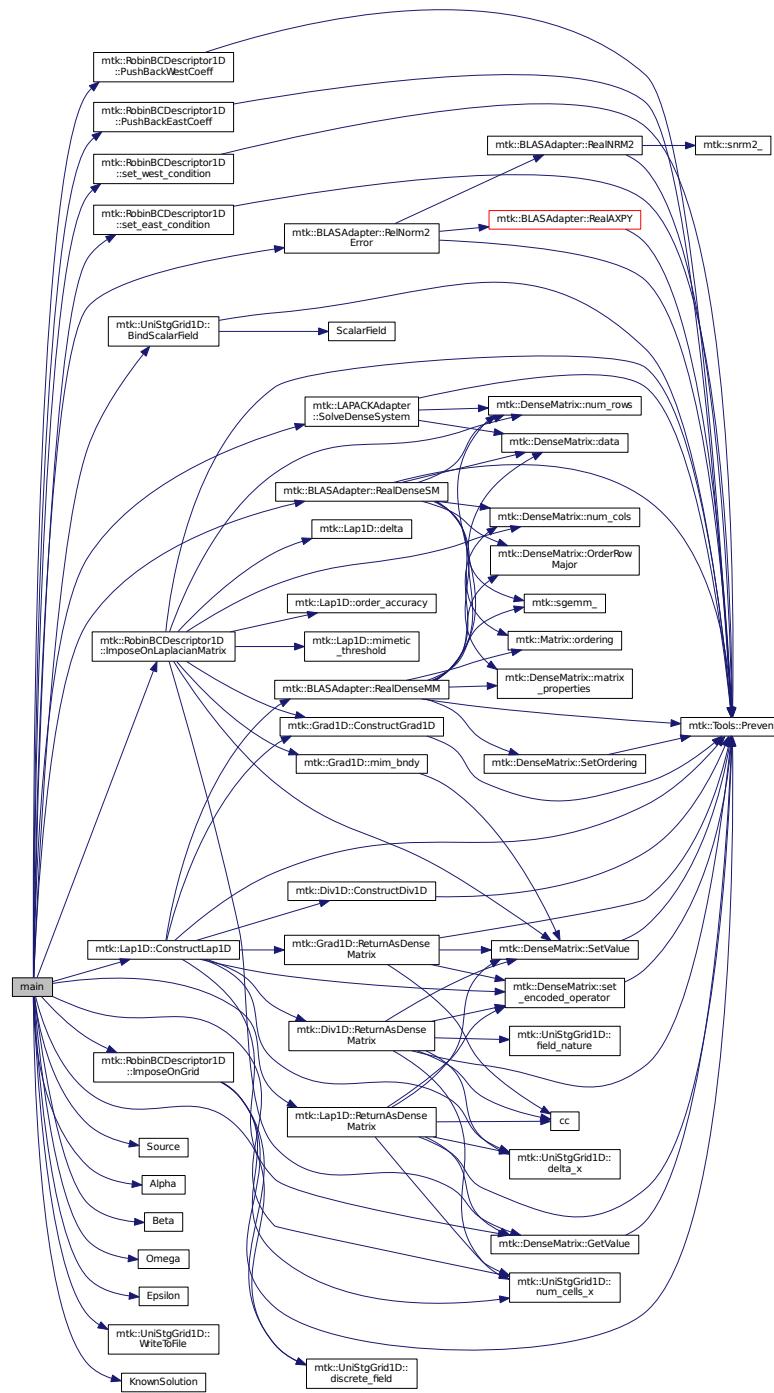
1. Discretize space.
2. Create mimetic operator as a matrix.

2.1. Multiply times -1 to mimic the problem.

1. Create grid for source term.
2. Apply Boundary Conditions to operator.
3. Apply Boundary Conditions to source term's grid.
4. Solve the problem.
5. Compare computed solution against known solution.

Definition at line 137 of file [1d_poisson.cc](#).

Here is the call graph for this function:



18.13.2.6 `mtk::RealOmega (const mtk::Real & tt)`

Definition at line 113 of file [1d_poisson.cc](#).

Here is the caller graph for this function:



18.13.2.7 mtk::Real Source (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 123 of file [1d_poisson.cc](#).

Here is the caller graph for this function:



18.14 1d_poisson.cc

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of

```

```

00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <cstdlib>
00090 #include <cmath>
00091
00092 #include <iostream>
00093 #include <fstream>
00094
00095 #include <vector>
00096
00097 #include "mtk.h"
00098
00099 mtk::Real Alpha(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00100     mtk::Real lambda{-1.0};
00101
00102     return -exp(lambda);
00103 }
00104
00105
00106 mtk::Real Beta(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00107
00108     mtk::Real lambda{-1.0};
00109
00110     return (exp(lambda) - 1.0)/lambda;
00111 }
00112
00113 mtk::Real Omega(const mtk::Real &tt) {
00114
00115     return -1.0;
00116 }
00117
00118 mtk::Real Epsilon(const mtk::Real &tt) {
00119
00120     return 0.0;
00121 }
00122
00123 mtk::Real Source(const mtk::Real &xx, const std::vector<mtk::Real> &pp) {
00124
00125     mtk::Real lambda{-1.0};
00126
00127     return -lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00128 }
00129
00130 mtk::Real KnownSolution(const mtk::Real &xx, const std::vector<mtk::Real> &
00131 pp) {
00132
00133     mtk::Real lambda{-1.0};
00134
00135     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00136 }
00137 int main () {
00138
00139     std::cout << "Example: Poisson Equation with Robin BCs on a";
00140     std::cout << " 1D Uniform Staggered Grid." << std::endl;
00141
00142
00143     mtk::Real west_bndy_x{0.0};
00144     mtk::Real east_bndy_x{1.0};
00145     int num_cells_x{10};
00146
00147     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00148
00149
00150     mtk::Lap1D lap;
00151
00152     if (!lap.ConstructLap1D()) {
00153
00154

```

```

00155     std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00156     return EXIT_FAILURE;
00157 }
00158
00159 std::cout << "lap=" << std::endl;
00160 std::cout << lap << std::endl;
00161
00162 mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00163
00164 std::cout << "lapm =" << std::endl;
00165 std::cout << lapm << std::endl;
00166
00167
00168 lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00169
00170 std::cout << "-lapm =" << std::endl;
00171 std::cout << lapm << std::endl;
00172
00173
00174 mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00175
00176 const std::vector<mtk::Real> lambda{ {-mtk::kOne} };
00177
00178 source.BindScalarField(Source, lambda);
00179
00180 std::cout << "source =" << std::endl;
00181 std::cout << source << std::endl;
00182
00183
00184 mtk::RobinBCDescriptor1D robin_bc_desc_1d;
00185
00186 robin_bc_desc_1d.PushBackWestCoeff(Alpha);
00187 robin_bc_desc_1d.PushBackWestCoeff(Beta);
00188
00189 robin_bc_desc_1d.PushBackEastCoeff(Alpha);
00190 robin_bc_desc_1d.PushBackEastCoeff(Beta);
00191
00192 robin_bc_desc_1d.set_west_condition(Omega);
00193 robin_bc_desc_1d.set_east_condition(Epsilon);
00194
00195 if (!robin_bc_desc_1d.ImposeOnLaplacianMatrix(lap, lapm, lambda)) {
00196     std::cerr << "BCs could not be bound to the matrix." << std::endl;
00197     return EXIT_FAILURE;
00198 }
00199
00200 std::cout << "Mimetic Laplacian operator with imposed BCs:" << std::endl;
00201 std::cout << lapm << std::endl;
00202
00203 if (!lapm.WriteToFile("1d_poisson_lapm.dat")) {
00204     std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00205     return EXIT_FAILURE;
00206 }
00207
00208
00209 robin_bc_desc_1d.ImposeOnGrid(source);
00210
00211
00212 std::cout << "source =" << std::endl;
00213 std::cout << source << std::endl;
00214
00215 if (!source.WriteToFile("1d_poisson_source.dat", "x", "s(x)")) {
00216     std::cerr << "Source term could not be written to disk." << std::endl;
00217     return EXIT_FAILURE;
00218 }
00219
00220
00221
00222
00223
00224 int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00225
00226
00227 if (!info) {
00228     std::cout << "System solved." << std::endl;
00229     std::cout << std::endl;
00230 } else {
00231     std::cerr << "Something wrong solving system! info = " << info << std::endl;
00232     std::cerr << "Exiting..." << std::endl;
00233     return EXIT_FAILURE;
00234 }
00235
00236 std::cout << "Computed solution:" << std::endl;
00237 std::cout << source << std::endl;
00238
00239 if (!source.WriteToFile("1d_poisson_comp_sol.dat", "x", "u(x)")) {
00240     std::cerr << "Solution could not be written to file." << std::endl;

```

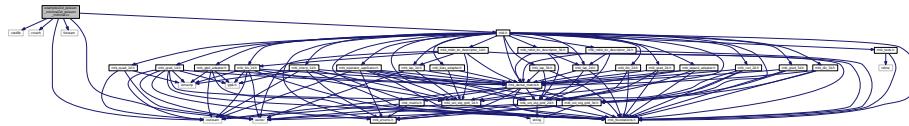
```

00241     return EXIT_FAILURE;
00242 }
00243
00245
00246 mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00247
00248 known_sol.BindScalarField(KnownSolution, std::vector<mtk::Real>());
00249
00250 std::cout << "known_sol =" << std::endl;
00251 std::cout << known_sol << std::endl;
00252
00253 if (!known_sol.WriteToFile("1d_poisson_known_sol.dat", "x", "u(x)")) {
00254     std::cerr << "Known solution could not be written to file." << std::endl;
00255     return EXIT_FAILURE;
00256 }
00257
00258 mtk::Real relative_norm_2_error{};
00259
00260 relative_norm_2_error =
00261     mtk::BLASAdapter::RelNorm2Error(source.
00262     discrete_field(),
00263             known_sol.discrete_field(),
00264             known_sol.num_cells_x());
00265
00266 std::cout << "relative_norm_2_error = ";
00267 std::cout << relative_norm_2_error << std::endl;
00268 }
```

18.15 examples/1d_poisson_minimal/1d_poisson_minimal.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <vector>
#include "mtk.h"
Include dependency graph for 1d_poisson_minimal.cc:
```



Functions

- `mtk::Real Alpha (const mtk::Real &tt, const std::vector< mtk::Real > &pp)`
- `mtk::Real Beta (const mtk::Real &tt, const std::vector< mtk::Real > &pp)`
- `mtk::Real Omega (const mtk::Real &tt)`
- `mtk::Real Epsilon (const mtk::Real &tt)`
- `mtk::Real Source (const mtk::Real &xx, const std::vector< mtk::Real > &pp)`
- `mtk::Real KnownSolution (const mtk::Real &xx, const std::vector< mtk::Real > &pp)`
- `int main ()`

18.15.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for $x \in \Omega = [a, b] = [0, 1]$.

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where $\lambda = -1$ is a real-valued parameter.

We consider Robin's boundary conditions of the form:

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

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

where $\alpha = -\exp(\lambda)$, $\beta = (\exp(\lambda) - 1.0)/\lambda$, $\omega = -1$, and $\varepsilon = 0$.

The analytical solution for this problem is given by:

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

The mimetic counterpart of this equation is:

$$-\tilde{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

Definition in file [1d_poisson_minimal.cc](#).

18.15.2 Function Documentation

18.15.2.1 mtk::Real Alpha (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line 97 of file [1d_poisson_minimal.cc](#).

Here is the caller graph for this function:



18.15.2.2 mtk::Real Beta (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line 102 of file [1d_poisson_minimal.cc](#).

Here is the caller graph for this function:



18.15.2.3 mtk::Real Epsilon (const mtk::Real & tt)

Definition at line 109 of file [1d_poisson_minimal.cc](#).

Here is the caller graph for this function:



18.15.2.4 mtk::Real KnownSolution (const mtk::Real & xx, const std::vector<mtk::Real> & pp)

Definition at line 116 of file [1d_poisson_minimal.cc](#).

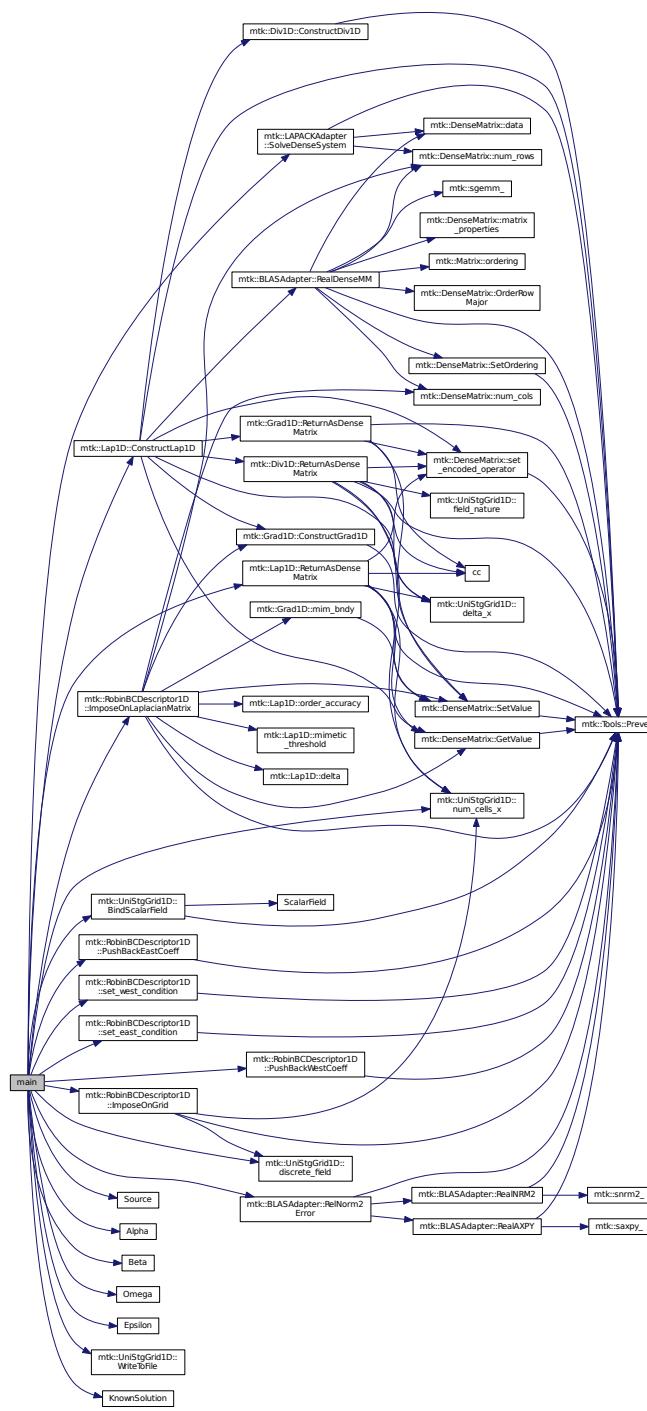
Here is the caller graph for this function:



18.15.2.5 int main ()

Definition at line 121 of file [1d_poisson_minimal.cc](#).

Here is the call graph for this function:



18.15.2.6 `mtk::RealOmega (const mtk::Real & tt)`

Definition at line 107 of file [1d_poisson_minimal.cc](#).

Here is the caller graph for this function:



18.15.2.7 mtk::Real Source (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 111 of file [1d_poisson_minimal.cc](#).

Here is the caller graph for this function:



18.16 1d_poisson_minimal.cc

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
  
```

```

00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <cstdlib>
00090 #include <cmath>
00091 #include <iostream>
00092 #include <fstream>
00093 #include <vector>
00094
00095 #include "mtk.h"
00096
00097 mtk::Real Alpha(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00098     mtk::Real lambda = -1.0;
00099     return -exp(lambda);
00100 }
00101
00102 mtk::Real Beta(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00103     mtk::Real lambda = -1.0;
00104     return (exp(lambda) - 1.0)/lambda;
00105 };
00106
00107 mtk::Real Omega(const mtk::Real &tt) { return -1.0; };
00108
00109 mtk::Real Epsilon(const mtk::Real &tt) { return 0.0; };
00110
00111 mtk::Real Source(const mtk::Real &xx, const std::vector<mtk::Real> &pp) {
00112     mtk::Real lambda = -1.0;
00113     return lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00114 }
00115
00116 mtk::Real KnownSolution(const mtk::Real &xx, const std::vector<mtk::Real> &
00117 pp) {
00118     mtk::Real lambda = -1.0;
00119     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00120 }
00121 int main () {
00122
00123     mtk::Real west_bndy_x{};
00124     mtk::Real east_bndy_x{1.0};
00125     int num_cells_x{5};
00126     mtk::Lap1D lap;
00127     if (!lap.ConstructLap1D()) {
00128         return EXIT_FAILURE;
00129     }
00130     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00131     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00132     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00133     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00134     source.BindScalarField(Source, std::vector<mtk::Real>());
00135     mtk::RobinBCDescriptor1D bcs;
00136     bcs.PushBackWestCoeff(Alpha);
00137     bcs.PushBackWestCoeff(Beta);
00138     bcs.PushBackEastCoeff(Alpha);
00139     bcs.PushBackEastCoeff(Beta);
00140     bcs.set_west_condition(Omega);
00141     bcs.set_east_condition(Epsilon);
00142     const std::vector<mtk::Real> lambda{ {-mtk::kOne} };
00143     if (!bcs.ImposeOnLaplacianMatrix(lap, lapm, lambda)) {
00144         return EXIT_FAILURE;
00145     }
00146     bcs.ImposeOnGrid(source);
00147     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00148     if (info != 0) {
00149         return EXIT_FAILURE;
00150     }
00151     source.WriteToFile("1d_poisson_minimal_comp_sol.dat", "x", "~u(x)");
00152     known_sol.BindScalarField(KnownSolution, std::vector<mtk::Real>());

```

```

00153     known_sol.WriteToFile("1d_poisson_minimal_known_sol.dat", "x", "u(x)");
00154     mtk::Real relative_norm_2_error =
00155         mtk::BLASAdapter::RelNorm2Error(source.
00156             discrete_field(),
00157             known_sol.discrete_field(),
00158             known_sol.num_cells_x());
00159     std::cout << relative_norm_2_error << std::endl;
00159 }

```

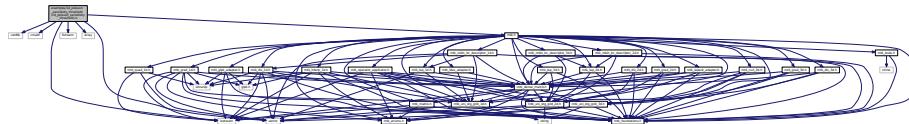
18.17 examples/1d_poisson_sensitivity_threshold/1d_poisson_sensitivity_threshold.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```

#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <array>
#include <vector>
#include "mtk.h"
Include dependency graph for 1d_poisson_sensitivity_threshold.cc:

```



Functions

- `mtk::Real Alpha (const mtk::Real &tt, const std::vector< mtk::Real > &pp)`
- `mtk::Real Beta (const mtk::Real &tt, const std::vector< mtk::Real > &pp)`
- `mtk::Real Omega (const mtk::Real &tt)`
- `mtk::Real Epsilon (const mtk::Real &tt)`
- `mtk::Real Source (const mtk::Real &xx, const std::vector< mtk::Real > &pp)`
- `mtk::Real KnownSolution (const mtk::Real &xx, const std::vector< mtk::Real > &pp)`
- `int main ()`

18.17.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for $x \in \Omega = [a, b] = [0, 1]$.

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where $\lambda = -1$ is a real-valued parameter.

We consider Robin's boundary conditions of the form:

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

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

where $\alpha = -\exp(\lambda)$, $\beta = \lambda^{-1}(\exp(\lambda) - 1.0)$, $\omega = -1$, and $\varepsilon = 0$.

The analytical solution for this problem is given by:

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

The mimetic counterpart of this equation is:

$$-\check{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

Definition in file [1d_poisson_sensitivity_threshold.cc](#).

18.17.2 Function Documentation

18.17.2.1 mtk::Real Alpha (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line 100 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.17.2.2 mtk::Real Beta (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line 107 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.17.2.3 mtk::Real Epsilon (const mtk::Real & tt)

Definition at line 119 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.17.2.4 mtk::Real KnownSolution (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 131 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.17.2.5 int main ()

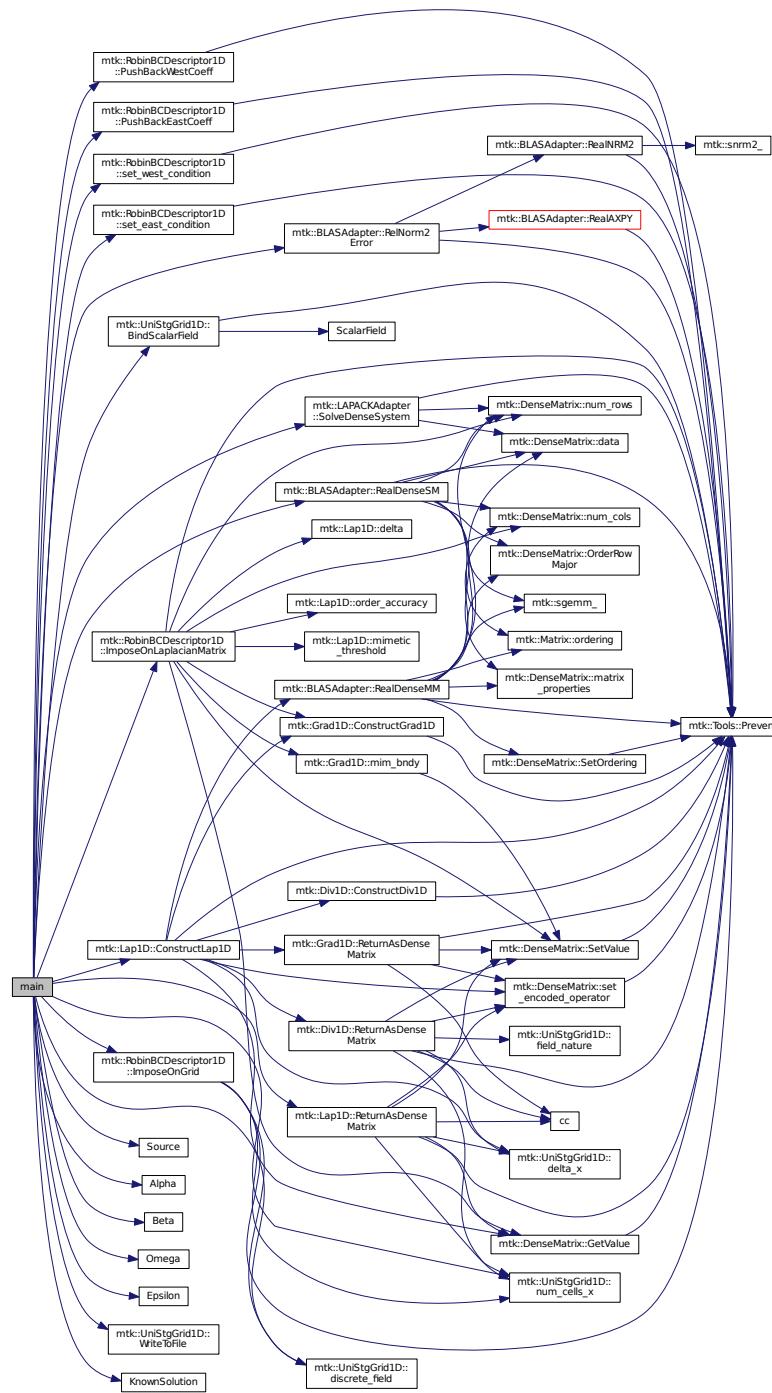
1. Discretize space.
2. Create mimetic operator as a matrix.

2.1. Multiply times -1 to mimic the problem.

1. Create grid for source term.
2. Apply Boundary Conditions to operator.
3. Apply Boundary Conditions to source term's grid.
4. Solve the problem.
5. Compare computed solution against known solution.

Definition at line 138 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the call graph for this function:



18.17.2.6 `mtk::Real Omega (const mtk::Real & tt)`

Definition at line 114 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.17.2.7 mtk::Real Source (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 124 of file [1d_poisson_sensitivity_threshold.cc](#).

Here is the caller graph for this function:



18.18 1d_poisson_sensitivity_threshold.cc

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
  
```

```

00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <cstdlib>
00090 #include <cmath>
00091
00092 #include <iostream>
00093 #include <fstream>
00094
00095 #include <array>
00096 #include <vector>
00097
00098 #include "mtk.h"
00099
00100 mtk::Real Alpha(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00101     mtk::Real lambda{-7.0};
00103
00104     return -exp(lambda);
00105 }
00106
00107 mtk::Real Beta(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00108
00109     mtk::Real lambda{-7.0};
00110
00111     return (exp(lambda) - 1.0)/lambda;
00112 };
00113
00114 mtk::Real Omega(const mtk::Real &tt) {
00115
00116     return -1.0;
00117 };
00118
00119 mtk::Real Epsilon(const mtk::Real &tt) {
00120
00121     return 0.0;
00122 };
00123
00124 mtk::Real Source(const mtk::Real &xx, const std::vector<mtk::Real> &pp) {
00125
00126     mtk::Real lambda{-7.0};
00127
00128     return -lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00129 }
00130
00131 mtk::Real KnownSolution(const mtk::Real &xx, const std::vector<mtk::Real> &
00132 pp) {
00133     mtk::Real lambda{-7.0};
00134
00135     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00136 }
00137
00138 int main () {
00139
00140     std::cout << "Example: Poisson Equation with Robin BCs on a";
00141     std::cout << " 1D Uniform Staggered Grid." << std::endl;
00142
00143     const int num_samples{10};
00144
00145     std::vector<mtk::Real> thresholds(num_samples);
00146
00147     thresholds.at(0) = mtk::kZero + mtk::kDefaultTolerance;
00148     for (size_t ii = 1; ii < thresholds.size(); ++ii) {
00149         thresholds.at(ii) = thresholds.at(ii - 1) + (mtk::Real) 0.1;
00150     }
00151
00152     std::array<int, 4> orders{ {8, 10, 12, 14} };

```

```

00153     int max_order{14};
00155
00157
00158     mtk::Real west_bndy_x{0.0};
00159     mtk::Real east_bndy_x{1.0};
00160     int num_cells_x{3*max_order - 1};
00161
00162     std::ofstream output_dat_file; // Output file.
00163
00164     output_dat_file.open("poisson_sensitivity_threshold.tex");
00165
00166     if (!output_dat_file.is_open()) {
00167         std::cerr << "Could not open data file." << std::endl;
00168         return EXIT_FAILURE;
00169     }
00170
00171     output_dat_file << "\\begin{tabular}[c]{cccc}" << std::endl;
00172     output_dat_file << "\\toprule" << std::endl;
00173 //    output_dat_file << "$k$ & \\multicolumn{4}{c}{Relative error} \\"\\\" <<
00174 //    std::endl;
00175 //    output_dat_file << " & $\\lambda = -1\$ & $\\lambda = -3\$"
00176 //    " & $\\lambda = -5\$ & $\\lambda = -7\$ \\"\\\" <<
00177 //    std::endl;
00178     output_dat_file << " & \\multicolumn{4}{c}{Relative error} \\"\\\" << std::endl;
00179     output_dat_file <<
00180     "$\\epsilon & k = 8\$ & k = 10\$ & k = 12\$ & k = 14\$ \\"\\\" << std::endl;
00181     output_dat_file << "\\midrule" << std::endl;
00182
00183     for (const mtk::Real &threshold: thresholds) {
00184
00185         output_dat_file << threshold;
00186
00187         for (const int &order: orders) {
00188
00189             mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00190
00192
00193             mtk::Lap1D lap;
00194
00195             if (!lap.ConstructLap1D(order, threshold)) {
00196                 std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00197                 return EXIT_FAILURE;
00198             }
00199
00200             std::cout << "lap=" << std::endl;
00201             std::cout << lap << std::endl;
00202
00203             mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00204
00205             std::cout << "lapm =" << std::endl;
00206             std::cout << lapm << std::endl;
00207
00208
00209             lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00210
00211             std::cout << "-lapm =" << std::endl;
00212             std::cout << lapm << std::endl;
00213
00214
00216
00217     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00218
00219     const std::vector<mtk::Real> lambda{ {-mtk::kOne} };
00220
00221     source.BindScalarField(Source, lambda);
00222
00223     std::cout << "source =" << std::endl;
00224     std::cout << source << std::endl;
00225
00226
00228     mtk::RobinBCDescriptor1D robin_bc_desc_1d;
00229
00230     robin_bc_desc_1d.PushBackWestCoeff(Alpha);
00231     robin_bc_desc_1d.PushBackWestCoeff(Beta);
00232
00233     robin_bc_desc_1d.PushBackEastCoeff(Alpha);
00234     robin_bc_desc_1d.PushBackEastCoeff(Beta);
00235
00236     robin_bc_desc_1d.set_west_condition(Omega);
00237     robin_bc_desc_1d.set_east_condition(Epsilon);
00238

```

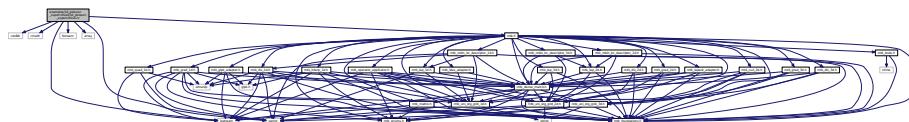
```

00239     if (!robin_bc_desc_1d.ImposeOnLaplacianMatrix(lap, lapm, lambda)) {
00240         std::cerr << "BCs could not be bound to the matrix." << std::endl;
00241         return EXIT_FAILURE;
00242     }
00243
00244     std::cout << "Mimetic Laplacian operator with imposed BCs:" << std::endl;
00245     std::cout << lapm << std::endl;
00246
00247     if (!lapm.WriteToFile("1d_poisson_lapm.dat")) {
00248         std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00249         return EXIT_FAILURE;
00250     }
00251
00252
00253     robin_bc_desc_1d.ImposeOnGrid(source);
00254
00255     std::cout << "source =" << std::endl;
00256     std::cout << source << std::endl;
00257
00258     if (!source.WriteToFile("1d_poisson_source.dat", "x", "s(x)")) {
00259         std::cerr << "Source term could not be written to disk." << std::endl;
00260         return EXIT_FAILURE;
00261     }
00262
00263
00264     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00265
00266     if (!info) {
00267         std::cout << "System solved." << std::endl;
00268         std::cout << std::endl;
00269     } else {
00270         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00271         std::cerr << "Exiting..." << std::endl;
00272         return EXIT_FAILURE;
00273     }
00274
00275
00276     std::cout << "Computed solution:" << std::endl;
00277     std::cout << source << std::endl;
00278
00279     if (!source.WriteToFile("1d_poisson_comp_sol.dat", "x", "~u(x)")) {
00280         std::cerr << "Solution could not be written to file." << std::endl;
00281         return EXIT_FAILURE;
00282     }
00283
00284
00285     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00286
00287     known_sol.BindScalarField(KnownSolution, std::vector<mtk::Real>());
00288
00289     std::cout << "known_sol =" << std::endl;
00290     std::cout << known_sol << std::endl;
00291
00292     if (!known_sol.WriteToFile("1d_poisson_known_sol.dat", "x", "u(x)")) {
00293         std::cerr << "Known solution could not be written to file." << std::endl;
00294         return EXIT_FAILURE;
00295     }
00296
00297     mtk::Real relative_norm_2_error{};
00298
00299     relative_norm_2_error =
00300         mtk::BLASAdapter::RelNorm2Error(source.
00301         discrete_field(),
00302                                         known_sol.discrete_field(),
00303                                         known_sol.num_cells_x());
00304
00305
00306     std::cout << "relative_norm_2_error = ";
00307     std::cout << relative_norm_2_error << std::endl;
00308
00309     output_dat_file << " & " << relative_norm_2_error;
00310 }
00311     output_dat_file << " \\\\" << std::endl;
00312 }
00313
00314     output_dat_file << "\\\bottomrule" << std::endl;
00315     output_dat_file << "\\\end{tabular}" << std::endl;
00316
00317     output_dat_file.close();
00318 }
```

18.19 examples/1d_poisson_supercritical/1d_poisson_supercritical.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <array>
#include <vector>
#include "mtk.h"
Include dependency graph for 1d_poisson_supercritical.cc:
```



Functions

- mtk::Real Alpha (const mtk::Real &tt, const std::vector< mtk::Real > &pp)
- mtk::Real Beta (const mtk::Real &tt, const std::vector< mtk::Real > &pp)
- mtk::Real Omega (const mtk::Real &tt)
- mtk::Real Epsilon (const mtk::Real &tt)
- mtk::Real Source (const mtk::Real &xx, const std::vector< mtk::Real > &pp)
- mtk::Real KnownSolution (const mtk::Real &xx, const std::vector< mtk::Real > &pp)
- int main ()

18.19.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for $x \in \Omega = [a, b] = [0, 1]$.

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where $\lambda = -1$ is a real-valued parameter.

We consider Robin's boundary conditions of the form:

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

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

where $\alpha = -\exp(\lambda)$, $\beta = \lambda^{-1}(\exp(\lambda) - 1.0)$, $\omega = -1$, and $\varepsilon = 0$.

The analytical solution for this problem is given by:

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

The mimetic counterpart of this equation is:

$$-\check{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

Definition in file [1d_poisson_supercritical.cc](#).

18.19.2 Function Documentation

18.19.2.1 mtk::Real Alpha (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line [100](#) of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.19.2.2 mtk::Real Beta (const mtk::Real & tt, const std::vector< mtk::Real > & pp)

Definition at line [107](#) of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.19.2.3 mtk::Real Epsilon (const mtk::Real & tt)

Definition at line [119](#) of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.19.2.4 mtk::Real KnownSolution (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 131 of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.19.2.5 int main ()

1. Perform experiment for the gradient operator.

1. Discretize space.

2. Create mimetic operator as a matrix.

2.1. Multiply times -1 to mimic the problem.

1. Create grid for source term.

2. Apply Boundary Conditions to operator.

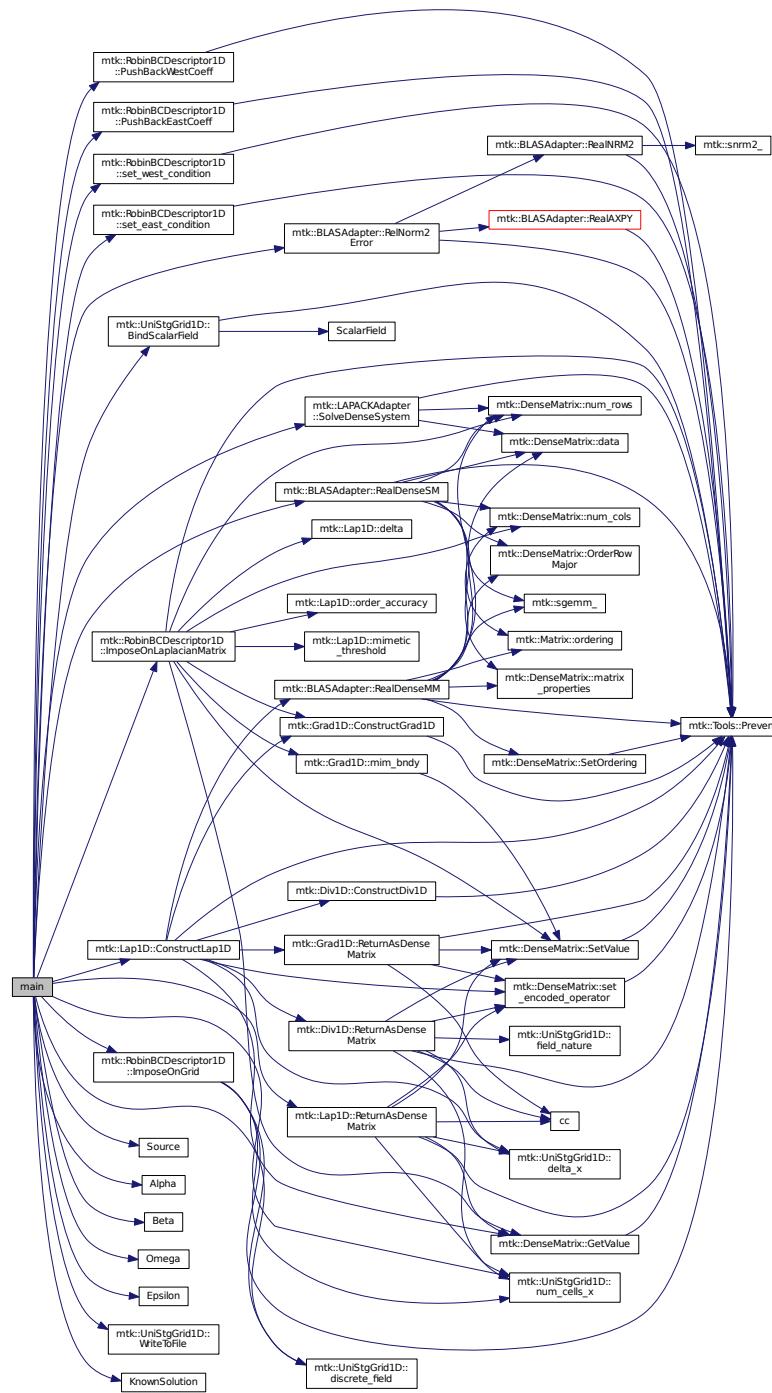
3. Apply Boundary Conditions to source term's grid.

4. Solve the problem.

5. Compare computed solution against known solution.

Definition at line 138 of file [1d_poisson_supercritical.cc](#).

Here is the call graph for this function:



18.19.2.6 `mtk::Real Omega (const mtk::Real & tt)`

Definition at line 114 of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.19.2.7 mtk::Real Source (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 124 of file [1d_poisson_supercritical.cc](#).

Here is the caller graph for this function:



18.20 1d_poisson_supercritical.cc

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
  
```

```

00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <cstdlib>
00090 #include <cmath>
00091
00092 #include <iostream>
00093 #include <fstream>
00094
00095 #include <array>
00096 #include <vector>
00097
00098 #include "mtk.h"
00099
00100 mtk::Real Alpha(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00101     mtk::Real lambda{pp.at(0)};
00103
00104     return -exp(lambda);
00105 }
00106
00107 mtk::Real Beta(const mtk::Real &tt, const std::vector<mtk::Real> &pp) {
00108
00109     mtk::Real lambda{pp.at(0)};
00110
00111     return (exp(lambda) - 1.0)/lambda;
00112 };
00113
00114 mtk::Real Omega(const mtk::Real &tt) {
00115
00116     return -1.0;
00117 };
00118
00119 mtk::Real Epsilon(const mtk::Real &tt) {
00120
00121     return 0.0;
00122 };
00123
00124 mtk::Real Source(const mtk::Real &xx, const std::vector<mtk::Real> &pp) {
00125
00126     mtk::Real lambda{pp.at(0)};
00127
00128     return -lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00129 }
00130
00131 mtk::Real KnownSolution(const mtk::Real &xx, const std::vector<mtk::Real> &
00132 pp) {
00133     mtk::Real lambda{pp.at(0)};
00134
00135     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00136 }
00137
00138 int main () {
00139
00140     std::cout << "Example: Poisson Equation with Robin BCs on a";
00141     std::cout << " 1D Uniform Staggered Grid." << std::endl;
00142
00143     mtk::Real west_bndy_x{0.0};
00144     mtk::Real east_bndy_x{1.0};
00145     std::array<int, 7> orders_accuracy{ {2, 4, 6, 8, 10, 12, 14} };
00146     std::array<mtk::Real, 4> lambda{ {-1.0, -3.0, -5.0, -7.0} };
00147     const int max_order{14};
00148     const int num_cells_x{3*max_order - 1};
00149
00150     std::ofstream output_dat_file; // Output file.
00151
00152
00153

```

```

00154     output_dat_file.open("rel_error.tex");
00155
00156     if (!output_dat_file.is_open()) {
00157         std::cerr << "Could not open data file." << std::endl;
00158         return EXIT_FAILURE;
00159     }
00160
00161     output_dat_file << "\\begin{tabular}[c]{c:cccc}" << std::endl;
00162     output_dat_file << "\\toprule" << std::endl;
00163     output_dat_file << "$k\$ & \\multicolumn{4}{c}{Relative error} \\\\" <<
00164         std::endl;
00165     output_dat_file << " & \$\\lambda = -1\$ & \$\\lambda = -3\$"
00166         " & \$\\lambda = -5\$ & \$\\lambda = -7\$ \\\\" <<
00167         std::endl;
00168     output_dat_file << "\\midrule" << std::endl;
00169
00170     for (const int &order: orders_accuracy) {
00171
00172         output_dat_file << order;
00173
00174         for (const mtk::Real &l1: lambda) {
00175
00176             mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00177
00178             mtk::Lap1D lap;
00179
00180             if (!lap.ConstructLap1D(order)) {
00181                 std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00182                 return EXIT_FAILURE;
00183             }
00184
00185             mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00186
00187             lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00188
00189             mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00190
00191             std::vector<mtk::Real> param{ {l1} };
00192
00193             source.BindScalarField(Source, param);
00194
00195             mtk::RobinBCDescriptor1D robin_bc_desc_1d;
00196
00197             robin_bc_desc_1d.PushBackWestCoeff(Alpha);
00198             robin_bc_desc_1d.PushBackWestCoeff(Beta);
00199
00200             robin_bc_desc_1d.PushBackEastCoeff(Alpha);
00201             robin_bc_desc_1d.PushBackEastCoeff(Beta);
00202
00203             robin_bc_desc_1d.set_west_condition(Omega);
00204             robin_bc_desc_1d.set_east_condition(Epsilon);
00205
00206             if (!robin_bc_desc_1d.ImposeOnLaplacianMatrix(lap, lapm, param)) {
00207                 std::cerr << "BCs could not be bound to the matrix." << std::endl;
00208                 return EXIT_FAILURE;
00209             }
00210
00211             if (!lapm.WriteToFile(
00212                 "1d_poisson_lapm_" +
00213                 std::to_string(order) +
00214                 std::to_string(l1) + ".dat")) {
00215                 std::cerr << "Laplacian matrix could not be written to disk." <<
00216                     std::endl;
00217                 return EXIT_FAILURE;
00218             }
00219
00220             robin_bc_desc_1d.ImposeOnGrid(source);
00221
00222             std::cout << "source =" << std::endl;
00223             std::cout << source << std::endl;
00224
00225             if (!source.WriteToFile("1d_poisson_source_" +
00226                 std::to_string(order) +
00227                 std::to_string(l1) + ".dat")) {
00228                 std::cerr << "Source term could not be written to disk." << std::endl;
00229
00230             }
00231
00232             robin_bc_desc_1d.ImposeOnGrid(source);
00233
00234             std::cout << "source =" << std::endl;
00235             std::cout << source << std::endl;
00236
00237             if (!source.WriteToFile("1d_poisson_source_" +
00238                 std::to_string(order) +
00239                 std::to_string(l1) + ".dat", "x", "s(x)")) {
00240                 std::cerr << "Source term could not be written to disk." << std::endl;
00241
00242             }
00243
00244         }
00245
00246     }
00247
00248 }
```

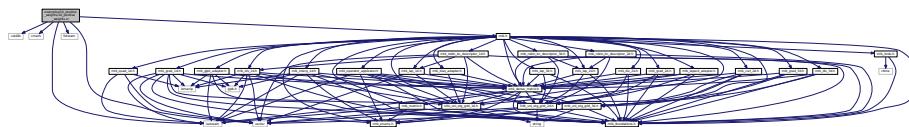
```

00241     return EXIT_FAILURE;
00242 }
00243
00245
00246     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00247
00248     if (!info) {
00249         std::cout << "System solved." << std::endl;
00250         std::cout << std::endl;
00251     } else {
00252         std::cerr << "Something wrong solving system! info = " << info <<
00253             std::endl;
00254         std::cerr << "Exiting..." << std::endl;
00255         return EXIT_FAILURE;
00256     }
00257
00258     if (!source.WriteToFile(
00259         "1d_poisson_comp_sol_" +
00260             std::to_string(order) +
00261             std::to_string(ll) +
00262             ".dat",
00263             "x",
00264             "~u(x)") {
00265         std::cerr << "Solution could not be written to file." << std::endl;
00266         return EXIT_FAILURE;
00267     }
00268
00269
00270     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00271
00272     known_sol.BindScalarField(KnownSolution, param);
00273
00274     if (!known_sol.WriteToFile(
00275         "1d_poisson_known_sol_" + std::to_string(order) +
00276             std::to_string(ll) +
00277             ".dat",
00278             "x",
00279             "u(x)") {
00280         std::cerr << "Known solution could not be written to file." <<
00281             std::endl;
00282         return EXIT_FAILURE;
00283     }
00284
00285     mtk::Real relative_norm_2_error{};
00286
00287     relative_norm_2_error =
00288         mtk::BLASAdapter::RelNorm2Error(source.
00289             discrete_field(),
00290                                         known_sol.discrete_field(),
00291                                         known_sol.num_cells_x());
00292
00293     std::cout << "order = ";
00294     std::cout << order << std::endl;
00295     std::cout << "relative_norm_2_error = ";
00296     std::cout << relative_norm_2_error << std::endl;
00297
00298     output_dat_file << " & " << relative_norm_2_error;
00299 }
00300     output_dat_file << "\\\\" << std::endl;
00301 }
00302
00303     output_dat_file << "\\bottomrule" << std::endl;
00304     output_dat_file << "\\end{tabular}" << std::endl;
00305
00306     output_dat_file.close();
00307 }
```

18.21 examples/1d_positive_weights/1d_positive_weights.cc File Reference

The CBS algorithm computes positive-definite weights, for 1D operators.

```
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <fstream>
#include <vector>
#include "mtk.h"
Include dependency graph for 1d_positive_weights.cc:
```



Functions

- int `main ()`

18.21.1 Detailed Description

Author

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

Definition in file [1d_positive_weights.cc](#).

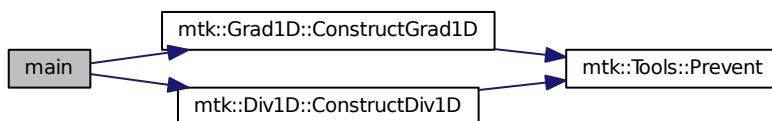
18.21.2 Function Documentation

18.21.2.1 int main ()

1. Create all critical-order divergence operators.
2. Create all critical-order divergence operators.

Definition at line 64 of file [1d_positive_weights.cc](#).

Here is the call graph for this function:



18.22 1d_positive_weights.cc

00001

```

00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cstdlib>
00055 #include <cmath>
00056
00057 #include <iostream>
00058 #include <fstream>
00059
00060 #include <vector>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Positive-Definite Weights for 1D Mimetic"
00067     "Operators." << std::endl;
00068
00069     mtk::Grad1D grad10;
00070
00071     bool assertion = grad10.ConstructGrad1D(10);
00072     if (!assertion) {
00073         std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00074         return EXIT_FAILURE;
00075     }
00076
00077     mtk::Grad1D grad12;
00078
00079     assertion = grad12.ConstructGrad1D(12);
00080     if (!assertion) {
00081         std::cerr << "Mimetic grad (12th order) could not be built." << std::endl;
00082         return EXIT_FAILURE;
00083     }
00084
00085     mtk::Grad1D grad14;
00086
00087     assertion = grad14.ConstructGrad1D(14);
00088
00089

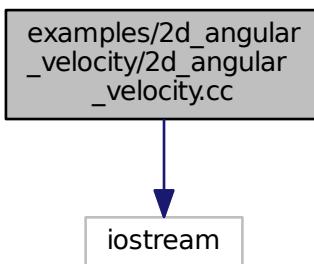
```

```
00090 if (!assertion) {
00091     std::cerr << "Mimetic grad (14th order) could not be built." << std::endl;
00092     return EXIT_FAILURE;
00093 }
00094
00095 mtk::Div1D div8;
00096
00097 assertion = div8.ConstructDiv1D(8);
00098 if (!assertion) {
00099     std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00100     return EXIT_FAILURE;
00101 }
00102
00103 mtk::Div1D div10;
00104
00105 assertion = div10.ConstructDiv1D(10);
00106 if (!assertion) {
00107     std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00108     return EXIT_FAILURE;
00109 }
00110
00111 mtk::Div1D div12;
00112
00113 assertion = div12.ConstructDiv1D(12);
00114 if (!assertion) {
00115     std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00116     return EXIT_FAILURE;
00117 }
00118
00119 mtk::Div1D div14;
00120
00121 assertion = div14.ConstructDiv1D(14);
00122 if (!assertion) {
00123     std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00124     return EXIT_FAILURE;
00125 }
00126
00127 }
00128 }
```

18.23 examples/2d.angular_velocity/2d.angular_velocity.cc File Reference

Compute the curl of a 2D angular velocity field.

```
#include <iostream>
Include dependency graph for 2d.angular_velocity.cc:
```



Functions

- int `main ()`

18.23.1 Detailed Description

We compute the curl of:

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

Author

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

Definition in file [2d_angular_velocity.cc](#).

18.23.2 Function Documentation

18.23.2.1 int main ()

Definition at line 106 of file [2d_angular_velocity.cc](#).

18.24 2d_angular_velocity.cc

```

00001
00013 /*
00014 Copyright (C) 2016, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES);

```

```

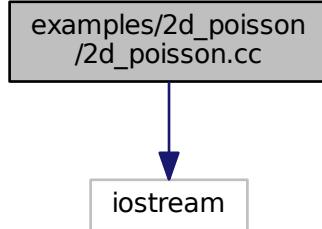
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #if __cplusplus == 201103L
00060
00061 #include <iostream>
00062 #include <fstream>
00063 #include <cmath>
00064
00065 #include <vector>
00066
00067 #include "mtk.h"
00068
00069 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const
00070   mtk::Real &yy) {
00071   return -yy;
00072 }
00073
00074 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
00075   mtk::Real &yy) {
00076   return xx;
00077 }
00078
00079 int main () {
00080
00081   std::cout << "Example: Curl of a angular velocity field." << std::endl;
00082
00083   mtk::Real aa = 0.0;
00084   mtk::Real bb = 4.0;
00085   mtk::Real cc = 0.0;
00086   mtk::Real dd = 4.0;
00087
00088   int nn = 10;
00089   int mm = 10;
00090
00091   mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm,
00092     mtk::FieldNature::VECTOR);
00093
00094   gg.BindVectorField(VectorFieldPComponent,
00095     VectorFieldQComponent);
00096
00097   if(!gg.WriteToFile("2d-angular_velocity_gg.dat", "x", "y", "v(x,y)")) {
00098     std::cerr << "Angular field could not be written to disk." << std::endl;
00099     return EXIT_FAILURE;
00100   }
00101
00102 #else
00103 #include <iostream>
00104 using std::cout;
00105 using std::endl;
00106 int main () {
00107   cout << "This code HAS to be compiled with support for C++11." << endl;
00108   cout << "Exiting..." << endl;
00109   return EXIT_SUCCESS;
00110 }
00111#endif

```

18.25 examples/2d_poisson/2d_poisson.cc File Reference

Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
Include dependency graph for 2d_poisson.cc:
```



Functions

- int `main ()`

18.25.1 Detailed Description

We solve:

$$\nabla^2 u(\mathbf{x}) = s(\mathbf{x}),$$

for $\mathbf{x} \in \Omega = [0, 1]^2$.

The source term function is defined as

$$s(x, y) = xye^{-0.5(x^2+y^2)}(x^2 + y^2 - 6).$$

Let $\partial\Omega = S \cup N \cup W \cup E$. We consider Dirichlet boundary conditions of the following form:

$$\forall \mathbf{x} \in W : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in E : u(1, y) = -e^{-0.5(1-y^2)}(1 - y^2).$$

$$\forall \mathbf{x} \in S : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in N : u(x, 1) = -e^{-0.5(x^2-1)}(x^2 - 1).$$

The analytical solution for this problem is given by

$$u(x, y) = xye^{-0.5(x^2+y^2)}.$$

Author

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

Definition in file [2d_poisson.cc](#).

18.25.2 Function Documentation

18.25.2.1 int main()

Definition at line 247 of file [2d_poisson.cc](#).

18.26 2d_poisson.cc

```

00001
00039 

```

```

00105     return mtk::kOne;
00106 }
00108
00109 mtk::Real WestBC(const mtk::Real &xx, const mtk::Real &tt) {
00110
00111     return mtk::kZero;
00112 }
00113
00114 mtk::Real EastBC(const mtk::Real &yy, const mtk::Real &tt) {
00115
00116     return yy*exp(-0.5*(mtk::kOne + yy*yy));
00117 }
00118
00119 mtk::Real SouthBC(const mtk::Real &xx, const mtk::Real &tt) {
00120
00121     return mtk::kZero;
00122 }
00123
00124 mtk::Real NorthBC(const mtk::Real &xx, const mtk::Real &tt) {
00125
00126     return xx*exp(-0.5*(xx*xx + mtk::kOne));
00127 }
00128
00129 mtk::Real KnownSolution(const mtk::Real &xx, const
00130     mtk::Real &yy) {
00131
00132     mtk::Real x_squared(xx*xx);
00133     mtk::Real y_squared(yy*yy);
00134     mtk::Real aux{-0.5*(x_squared + y_squared)};
00135
00136     return xx*yy*exp(aux);
00137 }
00138 int main () {
00139
00140     std::cout << "Example: Poisson Equation on a 2D Uniform Staggered Grid ";
00141     std::cout << "with Dirichlet and Neumann BCs." << std::endl;
00142
00143     mtk::Real west_bndy_x{0.0};
00144     mtk::Real east_bndy_x{1.0};
00145     mtk::Real south_bndy_y{0.0};
00146     mtk::Real north_bndy_y{1.0};
00147     int num_cells_x{10};
00148     int num_cells_y{10};
00149
00150     mtk::UniStgGrid2D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00151                                 south_bndy_y, north_bndy_y, num_cells_y);
00152
00153
00154     mtk::Lap2D lap;
00155
00156     if (!lap.ConstructLap2D(comp_sol)) {
00157         std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00158         return EXIT_FAILURE;
00159     }
00160
00161     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix());
00162
00163
00164     mtk::UniStgGrid2D source(west_bndy_x, east_bndy_x, num_cells_x,
00165                             south_bndy_y, north_bndy_y, num_cells_y);
00166
00167     source.BindScalarField(Source);
00168
00169
00170     mtk::RobinBCDescriptor2D bcd;
00171
00172
00173     bcd.PushBackWestCoeff(BCCoeff);
00174     bcd.PushBackEastCoeff(BCCoeff);
00175     bcd.PushBackSouthCoeff(BCCoeff);
00176     bcd.PushBackNorthCoeff(BCCoeff);
00177
00178     bcd.ImposeOnLaplacianMatrix(lap, comp_sol, lapm);
00179
00180     if (!lapm.WriteToFile("2d_poisson_lapm.dat")) {
00181         std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00182         return EXIT_FAILURE;
00183     }
00184
00185
00186 }
```

```

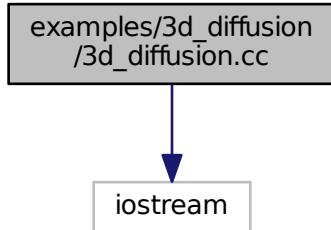
00190     bcd.set_west_condition(WestBC);
00191     bcd.set_east_condition(EastBC);
00192     bcd.set_south_condition(SouthBC);
00193     bcd.set_north_condition(NorthBC);
00194
00195     bcd.ImposeOnGrid(source);
00196
00197     if(!source.WriteToFile("2d_poisson_source.dat", "x", "y", "s(x,y)")) {
00198         std::cerr << "Source term could not be written to disk." << std::endl;
00199         return EXIT_FAILURE;
00200     }
00201
00203
00204     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00205
00206     if (!info) {
00207         std::cout << "System solved." << std::endl;
00208         std::cout << std::endl;
00209     } else {
00210         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00211         std::cerr << "Exiting..." << std::endl;
00212         return EXIT_FAILURE;
00213     }
00214
00215     if (!source.WriteToFile("2d_poisson_comp_sol.dat", "x", "y", "~u(x,y)")) {
00216         std::cerr << "Solution could not be written to file." << std::endl;
00217         return EXIT_FAILURE;
00218     }
00219
00220
00221     mtk::UniStgGrid2D known_sol(west_bndy_x, east_bndy_x, num_cells_x,
00222                                 south_bndy_y, north_bndy_y, num_cells_y);
00223
00224     known_sol.BindScalarField(KnownSolution);
00225
00226     if (!known_sol.WriteToFile("2d_poisson_known_sol.dat", "x", "y", "u(x,y)")) {
00227         std::cerr << "Known solution could not be written to file." << std::endl;
00228         return EXIT_FAILURE;
00229     }
00230
00231
00232     mtk::Real relative_norm_2_error{};
00233
00234     relative_norm_2_error =
00235         mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00236                                         known_sol.discrete_field(),
00237                                         known_sol.Size());
00238
00239     std::cout << "relative_norm_2_error = ";
00240     std::cout << relative_norm_2_error << std::endl;
00241 }
00242
00243 #else
00244 #include <iostream>
00245 using std::cout;
00246 using std::endl;
00247 int main () {
00248     cout << "This code HAS to be compiled with support for C++11." << endl;
00249     cout << "Exiting..." << endl;
00250     return EXIT_SUCCESS;
00251 }
00252 #endif

```

18.27 examples/3d_diffusion/3d_diffusion.cc File Reference

Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs.

```
#include <iostream>
Include dependency graph for 3d_diffusion.cc:
```



Functions

- int [main \(\)](#)

18.27.1 Detailed Description

We solve:

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{x}),$$

for $\mathbf{x} \in \Omega = [0, 1]^3$.

We consider autonomous homogeneous Dirichlet boundary conditions.

Author

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

Definition in file [3d_diffusion.cc](#).

18.27.2 Function Documentation

18.27.2.1 int main ()

Definition at line [139](#) of file [3d_diffusion.cc](#).

18.28 3d_diffusion.cc

```

00001
00016 /*
00017 Copyright (C) 2016, Computational Science Research Center, San Diego State
00018 University. All rights reserved.
00019
00020 Redistribution and use in source and binary forms, with or without modification,
00021 are permitted provided that the following conditions are met:
  
```

```
00022
00023 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00024 and a copy of the modified files should be reported once modifications are
00025 completed, unless these modifications are made through the project's GitHub
00026 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00027 should be developed and included in any deliverable.
00028
00029 2. Redistributions of source code must be done through direct
00030 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00031
00032 3. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 4. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders, and due credit should
00038 be given to the copyright holders.
00039
00040 5. Neither the name of the copyright holder nor the names of its contributors
00041 may be used to endorse or promote products derived from this software without
00042 specific prior written permission.
00043
00044 The copyright holders provide no reassurances that the source code provided does
00045 not infringe any patent, copyright, or any other intellectual property rights of
00046 third parties. The copyright holders disclaim any liability to any recipient for
00047 claims brought against recipient by any third party for infringement of that
00048 parties intellectual property rights.
00049
00050 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00051 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00052 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00053 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00054 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00055 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00056 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00057 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00058 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00059 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00060 */
00061
00062 #if __cplusplus == 201103L
00063
00064 #include <iostream>
00065 #include <fstream>
00066 #include <cmath>
00067
00068 #include <vector>
00069
00070 #include "mtk.h"
00071
00072 mtk::Real InitialCondition(const mtk::Real &xx,
00073                             const mtk::Real &yy,
00074                             const mtk::Real &zz) {
00075
00076     mtk::Real rr{0.3};
00077
00078     mtk::Real aux{xxxx + yy*yy + zz*zz};
00079
00080     return (aux < rr? rr - aux: mtk::kZero);
00081 }
00082
00083 int main () {
00084
00085     std::cout << "Example: Diffusion Equation in 3D "
00086         "with Dirichlet BCs." << std::endl;
00087
00088
00089     mtk::Real west_bndy_x{0.0};
00090     mtk::Real east_bndy_x{1.0};
00091     mtk::Real south_bndy_y{0.0};
00092     mtk::Real north_bndy_y{1.0};
00093     mtk::Real bottom_bndy_z{0.0};
00094     mtk::Real top_bndy_z{1.0};
00095
00096     int num_cells_x{10};
00097     int num_cells_y{10};
00098     int num_cells_z{10};
00099
00100     mtk::UniStgGrid3D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00101                                 south_bndy_y, north_bndy_y, num_cells_y,
00102                                 bottom_bndy_z, top_bndy_z, num_cells_z);
```

```

00104
00106
00107     comp_sol.BindScalarField(InitialCondition);
00108
00109     if(!comp_sol.WriteToFile("3d_diffusion_comp_sol.dat",
00110         "x",
00111         "y",
00112         "z",
00113         "Initial u(x,y,z)")) {
00114         std::cerr << "Error writing to file." << std::endl;
00115         return EXIT_FAILURE;
00116     }
00117
00119
00120     mtk::Lap3D lap;
00121
00122     if (!lap.ConstructLap3D(comp_sol)) {
00123         std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00124         return EXIT_FAILURE;
00125     }
00126
00127     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix());
00128
00129     if (!lapm.WriteToFile("3d_diffusion_lapm.dat")) {
00130         std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00131         return EXIT_FAILURE;
00132     }
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140     cout << "This code HAS to be compiled with support for C++11." << endl;
00141     cout << "Exiting..." << endl;
00142     return EXIT_SUCCESS;
00143 }
00144 #endif

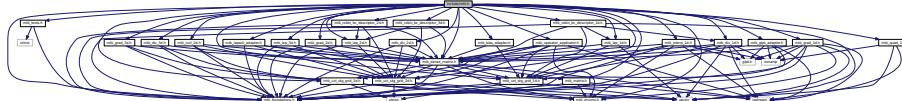
```

18.29 include/mtk.h File Reference

Includes the entire API.

```
#include "mtk_foundations.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_matrix.h"
#include "mtk_dense_matrix.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
#include "mtk_robin_bc_descriptor_1d.h"
#include "mtk_quad_1d.h"
#include "mtk_interp_1d.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"
#include "mtk_lap_2d.h"
#include "mtk_robin_bc_descriptor_2d.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"
#include "mtk_robin_bc_descriptor_3d.h"
#include "mtk_operator_applicator.h"
```

Include dependency graph for mtk.h:



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



18.29.1 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 define.

Author

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

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

18.30 mtk.h

```

00001
00015 /*
00016 Copyright (C) 2016, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00284 #ifndef MTK_INCLUDE_MTK_H_
00285 #define MTK_INCLUDE_MTK_H_
00286
00294 #include "mtk_foundations.h"
00295
00303 #include "mtk_enums.h"
00304
00312 #include "mtk_tools.h"
00313
00321 #include "mtk_matrix.h"
00322 #include "mtk_dense_matrix.h"
00323
00331 #include "mtk blas adapter.h"
00332 #include "mtk lapack adapter.h"
00333 #include "mtk glpk adapter.h"
00334
00342 #include "mtk uni stg grid 1d.h"
00343 #include "mtk uni stg grid 2d.h"
00344 #include "mtk uni stg grid 3d.h"
00345
00353 #include "mtk grad 1d.h"
00354 #include "mtk div 1d.h"

```

```

00355 #include "mtk_lap_1d.h"
00356 #include "mtk_robin_bc_descriptor_1d.h"
00357 #include "mtk_quad_1d.h"
00358 #include "mtk_interp_1d.h"
00359
00360 #include "mtk_grad_2d.h"
00361 #include "mtk_div_2d.h"
00362 #include "mtk_curl_2d.h"
00363 #include "mtk_lap_2d.h"
00364 #include "mtk_robin_bc_descriptor_2d.h"
00365
00366 #include "mtk_grad_3d.h"
00367 #include "mtk_div_3d.h"
00368 #include "mtk_foundations.h"
00369 #include "mtk_lap_3d.h"
00370 #include "mtk_robin_bc_descriptor_3d.h"
00371
00372 #include "mtk_operator_applicator.h"
00373
00374 #endif // End of: MTK_INCLUDE_MTK_H_

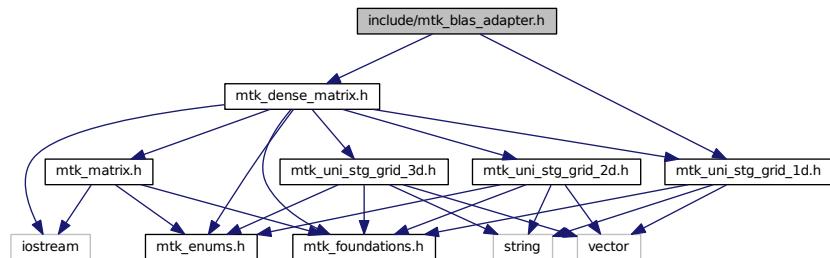
```

18.31 include/mtk_blas_adapter.h File Reference

Declaration of an adapter class for the BLAS API.

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

Include dependency graph for mtk_blas_adapter.h:



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



Classes

- class [mtk::BLASAdapter](#)
Adapter class for the BLAS API.

Namespaces

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

18.31.1 Detailed Description

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

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

The BLAS can be installed from links given in the See Also section of this page.

See also

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

Author

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

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

18.32 mtk blas adapter.h

```
00001
00025 /*
00026 Copyright (C) 2016, Computational Science Research Center, San Diego State
00027 University. All rights reserved.
00028
00029 Redistribution and use in source and binary forms, with or without modification,
00030 are permitted provided that the following conditions are met:
00031
00032 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00033 and a copy of the modified files should be reported once modifications are
00034 completed, unless these modifications are made through the project's GitHub
00035 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00036 should be developed and included in any deliverable.
00037
00038 2. Redistributions of source code must be done through direct
00039 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00040
00041 3. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 4. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders, and due credit should
00047 be given to the copyright holders.
00048
00049 5. Neither the name of the copyright holder nor the names of its contributors
00050 may be used to endorse or promote products derived from this software without
00051 specific prior written permission.
00052
00053 The copyright holders provide no reassurances that the source code provided does
00054 not infringe any patent, copyright, or any other intellectual property rights of
00055 third parties. The copyright holders disclaim any liability to any recipient for
00056 claims brought against recipient by any third party for infringement of that
00057 parties intellectual property rights.
00058
00059 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00060 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00061 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00062 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00063 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00064 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00065 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00066 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

```

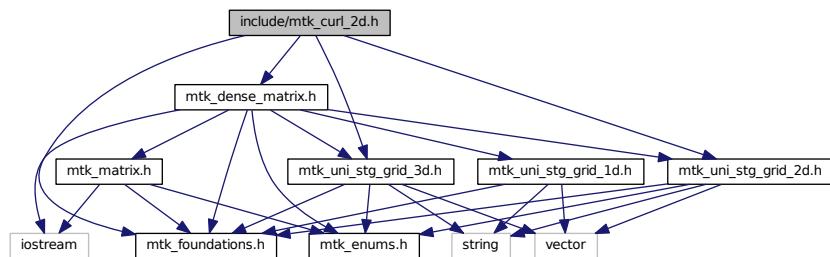
00067 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00068 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00069 */
00070
00071 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00072 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00073
00074 #include "mtk_dense_matrix.h"
00075 #include "mtk_uni_stg_grid_1d.h"
00076
00077 namespace mtk {
00078
00100 class BLASAdapter {
00101 public:
00110     static Real RealNRM2(Real *in, int &in_length);
00111
00128     static void RealAXPY(Real alpha, Real *xx, Real *yy, int &in_length);
00129
00144     static Real RelNorm2Error(Real *computed, Real *known, int length);
00145
00162     static void RealDenseMV(Real &alpha,
00163                             DenseMatrix &aa,
00164                             Real *xx,
00165                             Real &beta,
00166                             Real *yy);
00167
00188     static void RealDenseMV(Real &alpha,
00189                             Real &aa,
00190                             MatrixOrdering &ordering,
00191                             int num_rows,
00192                             int num_cols,
00193                             int lda,
00194                             Real *xx,
00195                             Real &beta,
00196                             Real *yy);
00197
00211     static DenseMatrix RealDenseMM(DenseMatrix &aa,
00212                                     DenseMatrix &bb);
00212
00227     static DenseMatrix RealDenseSM(Real alpha,
00228                                     DenseMatrix &aa);
00228 }
00229 }
00230#endif // End of: MTK_INCLUDE_BLAS_ADAPTER_H_

```

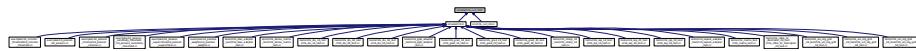
18.33 include/mtk_curl_2d.h File Reference

Includes the definition of the class Curl2D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"
Include dependency graph for mtk_curl_2d.h:
```



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



Classes

- class [mtk::Curl2D](#)

Implements a 2D mimetic curl operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.33.1 Detailed Description

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

Author

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

Definition in file [mtk_curl_2d.h](#).

18.34 mtk_curl_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that

```

```

00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_CURL_2D_H_
00058 #define MTK_INCLUDE_MTK_CURL_2D_H_
00059
00060 #include "mtk_foundations.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk{
00066
00077 class Curl2D {
00078 public:
00082     UniStgGrid3D operator*(const UniStgGrid2D &grid) const;
00083
00087     Curl2D();
00088
00094     Curl2D(const Curl2D &curl);
00095
00099     ~Curl2D();
00100
00106     bool ConstructCurl2D(const UniStgGrid2D &grid,
00107                           int order_accuracy = kDefaultOrderAccuracy,
00108                           Real mimetic_threshold = kDefaultMimeticThreshold);
00109
00115     DenseMatrix ReturnAsDenseMatrix() const;
00116
00117 private:
00118     DenseMatrix curl_;
00119
00120     int order_accuracy_;
00121
00122     Real mimetic_threshold_;
00123 };
00124 }
00125 #endif // End of: MTK_INCLUDE_MTK_CURL_2D_H_

```

18.35 include/mtk_dense_matrix.h File Reference

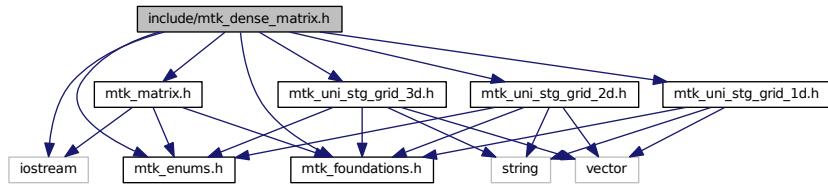
Declaration of a class dense matrix implemented using a 1D array.

```

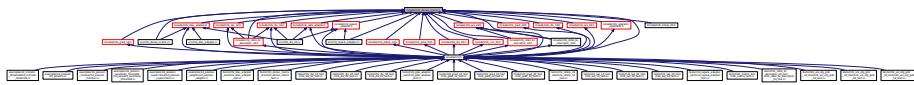
#include <iostream>
#include "mtk_foundations.h"
#include "mtk_enums.h"
#include "mtk_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for mtk_dense_matrix.h:



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



Classes

- class [mtk::DenseMatrix](#)
Defines a common dense matrix, using a 1D array.

Namespaces

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

18.35.1 Detailed Description

The construction of 1D mimetic operators exclusively involves dense matrix arithmetic. We encapsulate the complexity of a dense matrix in this class.

Author

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

Note

We prefer composition to inheritance [Reedy, 2011]. The main reason for this preference is that inheritance produces a more tightly coupled design. When a class inherits from another type be it public, protected, or private inheritance the subclass gains access to all public and protected members of the base class, whereas with composition, the class is only coupled to the public members of the other class. Furthermore, if you only hold a pointer to the other object, then your interface can use a forward declaration of the class rather than #include its full definition. This results in greater compile-time insulation and improves the time it takes to compile your code.

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

18.36 mtk_dense_matrix.h

```

00001
00021 /*
00022 Copyright (C) 2016, Computational Science Research Center, San Diego State
00023 University. All rights reserved.
00024
00025 Redistribution and use in source and binary forms, with or without modification,
00026 are permitted provided that the following conditions are met:
00027
00028 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00029 and a copy of the modified files should be reported once modifications are
00030 completed, unless these modifications are made through the project's GitHub
00031 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00032 should be developed and included in any deliverable.
00033
00034 2. Redistributions of source code must be done through direct
00035 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00036
00037 3. Redistributions in binary form must reproduce the above copyright notice,
00038 this list of conditions and the following disclaimer in the documentation and/or
00039 other materials provided with the distribution.
00040
00041 4. Usage of the binary form on proprietary applications shall require explicit
00042 prior written permission from the the copyright holders, and due credit should
00043 be given to the copyright holders.
00044
00045 5. Neither the name of the copyright holder nor the names of its contributors
00046 may be used to endorse or promote products derived from this software without
00047 specific prior written permission.
00048
00049 The copyright holders provide no reassurances that the source code provided does
00050 not infringe any patent, copyright, or any other intellectual property rights of
00051 third parties. The copyright holders disclaim any liability to any recipient for
00052 claims brought against recipient by any third party for infringement of that
00053 parties intellectual property rights.
00054
00055 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00056 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00057 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00058 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00059 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00060 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00061 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00062 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00063 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00064 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00065 */
00066
00067 #ifndef MTK_INCLUDE_DENSE_MATRIX_H_
00068 #define MTK_INCLUDE_DENSE_MATRIX_H_
00069
00070 #include <iostream>
00071
00072 #include "mtk_foundations.h"
00073 #include "mtk_enums.h"
00074 #include "mtk_matrix.h"
00075 #include "mtk_uni_stg_grid_1d.h"
00076 #include "mtk_uni_stg_grid_2d.h"
00077 #include "mtk_uni_stg_grid_3d.h"
00078
00079 namespace mtk {
00080
00093 class DenseMatrix {
00094 public:
00098     friend std::ostream& operator <<(std::ostream &stream, DenseMatrix &in);
00099
00107     DenseMatrix& operator =(const DenseMatrix &in);
00108
00112     bool operator ==(const DenseMatrix &in);
00113
00117     DenseMatrix();
00118
00124     DenseMatrix(const DenseMatrix &in);
00125
00134     DenseMatrix(const int &num_rows, const int &num_cols);
00135
00161     DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00162
00196     DenseMatrix(const Real *const gen,

```

```

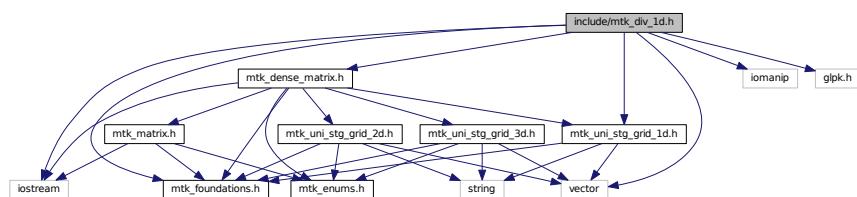
00197         const int &gen_length,
00198         const int &pro_length,
00199         const bool &transpose);
00200
00204     ~DenseMatrix();
00205
00211     EncodedOperator encoded_operator() const;
00212
00218     void set_encoded_operator(const EncodedOperator &op);
00219
00225     Matrix matrix_properties() const noexcept;
00226
00232     int num_rows() const noexcept;
00233
00239     int num_cols() const noexcept;
00240
00246     Real* data() const noexcept;
00247
00255     void SetOrdering(mtk::MatrixOrdering oo) noexcept;
00256
00265     Real GetValue(const int &row_coord, const int &col_coord) const noexcept;
00266
00274     void SetValue(const int &row_coord,
00275                   const int &col_coord,
00276                   const Real &val) noexcept;
00277
00281     void Transpose();
00282
00286     void OrderRowMajor();
00287
00291     void OrderColMajor();
00292
00303     static DenseMatrix Kron(const DenseMatrix &aa,
00304                             const DenseMatrix &bb);
00305
00315     bool WriteToFile(const std::string &filename) const;
00316
00317 private:
00318     Matrix matrix_properties_;
00319
00320     Real *data_;
00321 };
00322 }
00323 #endif // End of: MTK_INCLUDE_DENSE_MATRIX_H_

```

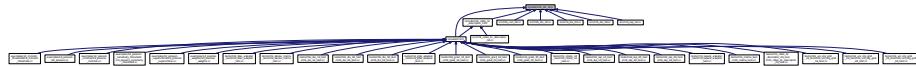
18.37 include/mtk_div_1d.h File Reference

Includes the definition of the class Div1D.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "glpk.h"
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
Include dependency graph for mtk_div_1d.h:
```



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



Classes

- class [mtk::Div1D](#)
Implements a 1D mimetic divergence operator.

Namespaces

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

18.37.1 Detailed Description

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

Author

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

Definition in file [mtk_div_1d.h](#).

18.38 mtk_div_1d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does

```

```

00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_DIV_1D_H_
00058 #define MTK_INCLUDE_DIV_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include <vector>
00064
00065 #include "glpk.h"
00066
00067 #include "mtk_foundations.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_1d.h"
00070
00071 namespace mtk {
00072
00083 class Div1D {
00084 public:
00088 friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00089
00093 Div1D();
00094
00100 Div1D(const Div1D &div);
00101
00105 ~Div1D();
00106
00112 bool ConstructDiv1D(int order_accuracy = kDefaultOrderAccuracy,
00113 Real mimetic_threshold = kDefaultMimeticThreshold);
00114
00120 int num_bndy_coeffs() const;
00121
00127 Real *coeffs_interior() const;
00128
00134 Real *weights_crs(void) const;
00135
00141 Real *weights_cbs(void) const;
00142
00148 int num_feasible_sols() const;
00149
00155 DenseMatrix mim_bndy() const;
00156
00162 std::vector<Real> sums_rows_mim_bndy() const;
00163
00169 Real mimetic_measure() const;
00170
00176 DenseMatrix ReturnAsDenseMatrix(const
UniStgGrid1D &grid) const;
00177
00183 DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
const;
00184
00185 private:
00191 bool ComputeStencilInteriorGrid(void);
00192
00199 bool ComputeRationalBasisNullSpace(void);
00200
00206 bool ComputePreliminaryApproximations(void);
00207
00213 bool ComputeWeights(void);
00214
00220 bool ComputeStencilBoundaryGrid(void);
00221
00227 bool AssembleOperator(void);
00228

```

```

00229 int order_accuracy_;
00230 int dim_null_;
00231 int num_bndy_coeffs_;
00232 int divergence_length_;
00233 int minrow_;
00234 int row_;
00235 int num_feasible_sols_;
00236
00237 DenseMatrix rat_basis_null_space_;
00238
00239 Real *coeffs_interior_;
00240 Real *prem_apps_;
00241 Real *weights_crs_;
00242 Real *weights_cbs_;
00243 Real *mim_bndy_;
00244 Real *divergence_;
00245
00246 Real mimetic_threshold_;
00247 Real mimetic_measure_;
00248
00249 std::vector<Real> sums_rows_mim_bndy_;
00250 };
00251 }
00252 #endif // End of: MTK_INCLUDE_DIV_1D_H_

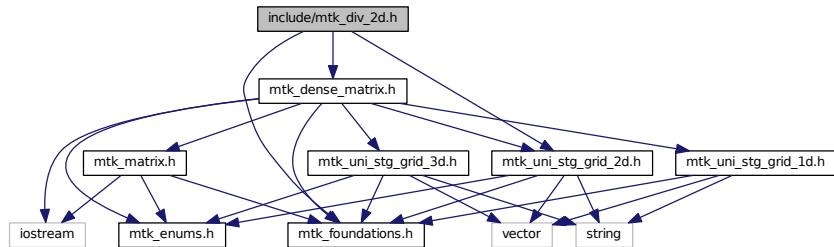
```

18.39 include/mtk_div_2d.h File Reference

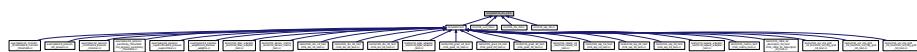
Includes the definition of the class Div2D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk_div_2d.h:



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



Classes

- class [mtk::Div2D](#)

Implements a 2D mimetic divergence operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.39.1 Detailed Description

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

Author

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

Definition in file [mtk_div_2d.h](#).

18.40 mtk_div_2d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_2D_H_
00058 #define MTK_INCLUDE_MTK_DIV_2D_H_
00059
00060 #include "mtk_foundations.h"

```

```

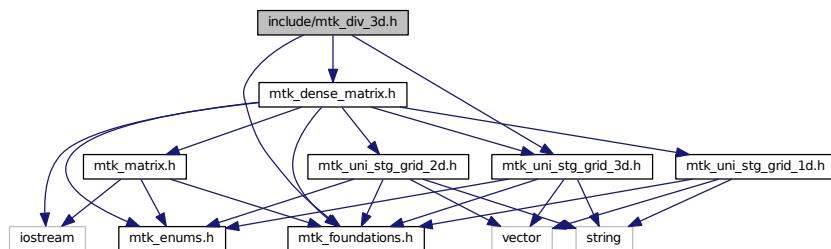
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Div2D {
00077 public:
00081     Div2D();
00082
00088     Div2D(const Div2D &div);
00089
00093     ~Div2D();
00094
00100     bool ConstructDiv2D(const UniStgGrid2D &grid,
00101                     int order_accuracy = kDefaultOrderAccuracy,
00102                     Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109     DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112     DenseMatrix divergence_;
00113
00114     int order_accuracy_;
00115
00116     Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_DIV_2D_H_

```

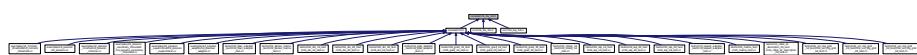
18.41 include/mtk_div_3d.h File Reference

Includes the definition of the class Div3D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"
Include dependency graph for mtk_div_3d.h:
```



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



Classes

- class `mtk::Div3D`

Implements a 3D mimetic divergence operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.41.1 Detailed Description

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

Author

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

Definition in file [mtk_div_3d.h](#).

18.42 mtk_div_3d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_3D_H_
00058 #define MTK_INCLUDE_MTK_DIV_3D_H_
00059
00060 #include "mtk_foundations.h"

```

```

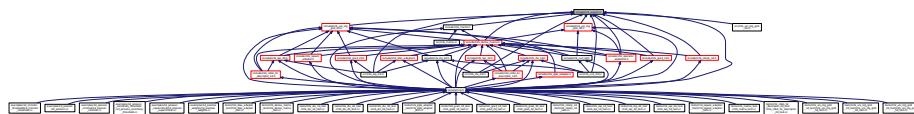
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{
00065
00076 class Div3D {
00077 public:
00081     Div3D();
00082
00088     Div3D(const Div3D &div);
00089
00093     ~Div3D();
00094
00100     bool ConstructDiv3D(const UniStgGrid3D &grid,
00101                     int order_accuracy = kDefaultOrderAccuracy,
00102                     Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109     DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112     DenseMatrix divergence_;
00113
00114     int order_accuracy_;
00115
00116     Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_DIV_3D_H_

```

18.43 include/mtk_enums.h File Reference

Definitions of the enumeration types in the MTK.

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



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Enumerations

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

- enum `mtk::EncodedOperator {`
`mtk::EncodedOperator::NOOP, mtk::EncodedOperator::GRADIENT, mtk::EncodedOperator::DIVERGENCE,`
`mtk::EncodedOperator::INTERPOLATION,`
`mtk::EncodedOperator::CURL, mtk::EncodedOperator::LAPLACIAN }`

Operators matrices can encode.

18.43.1 Detailed Description

Enumeration types are used throughout the MTK to differentiate instances of derived classes, as well as for mnemonic purposes. In this file, definitions for the enumeration types are listed alphabetically.

Author

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

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

18.44 mtk_enums.h

```

00001
00012 /*
00013 Copyright (C) 2016, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_ENUMS_H_
00059 #define MTK_INCLUDE_ENUMS_H_

```

```

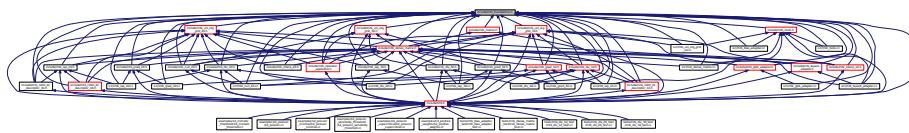
00060
00061 namespace mtk {
00062
00077 enum class MatrixStorage {
00078     DENSE,
00079     BANDED,
00080     CRS
00081 };
00082
00095 enum class MatrixOrdering {
00096     ROW_MAJOR,
00097     COL_MAJOR
00098 };
00099
00113 enum class FieldNature {
00114     SCALAR,
00115     VECTOR
00116 };
00117
00127 enum class DirInterp {
00128     SCALAR_TO_VECTOR,
00129     VECTOR_TO_SCALAR
00130 };
00131
00141 enum class EncodedOperator {
00142     NOOP,
00143     GRADIENT,
00144     DIVERGENCE,
00145     INTERPOLATION,
00146     CURL,
00147     LAPLACIAN
00148 };
00149 }
00150 #endif // End of: MTK_INCLUDE_ENUMS_H_

```

18.45 include/mtk_foundations.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}](#)

- const float `mtk::kOne` {1.0f}

MTK's one defined according to selective compilation.
- const float `mtk::kTwo` {2.0f}

MTK's two defined according to selective compilation.
- const float `mtk::kDefaultTolerance` {1e-7f}

Considered tolerance for comparisons in numerical methods.
- const float `mtk::kDefaultMimeticThreshold` {1e-6f}

Default threshold for higher-order mimetic operators.
- const int `mtk::kDefaultOrderAccuracy` {2}

Default order of accuracy for mimetic operators.
- const int `mtk::kCriticalOrderAccuracyGrad` {10}

At this order (and higher) we must use the CBSA to construct gradients.
- const int `mtk::kCriticalOrderAccuracyDiv` {8}

At this order (and higher) we must use the CBSA to construct divergences.

18.45.1 Detailed Description

This file contains the fundamental definitions that classes of the MTK rely on to be implemented. Examples of these definitions are the definition of fundamental data types, and global variables affecting the construction of mimetic operators, among others.

Author

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

Todo Test selective precision mechanisms.

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

18.46 mtk_foundations.h

```

00001
00015 /*
00016 Copyright (C) 2016, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors

```

```

00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_ROOTS_H_
00062 #define MTK_INCLUDE_ROOTS_H_
00063
00064 namespace mtk {
00065
00066 #ifdef MTK_PRECISION_DOUBLE
00067     typedef double Real;
00068 #else
00069     typedef float Real;
00070 #endif
00071
00072 #ifdef MTK_PRECISION_DOUBLE
00073     const double kZero{0.0};
00074     const double kOne{1.0};
00075     const double kTwo{2.0};
00076 #else
00077     const float kZero{0.0f};
00078     const float kOne{1.0f};
00079     const float kTwo{2.0f};
00080 #endif
00081
00082 #ifdef MTK_PRECISION_DOUBLE
00083     const double kDefaultTolerance{1e-7};
00084 #else
00085     const float kDefaultTolerance{1e-7f};
00086 #endif
00087
00088 #ifdef MTK_PRECISION_DOUBLE
00089     const double kDefaultMimeticThreshold{1e-6};
00090 #else
00091     const float kDefaultMimeticThreshold{1e-6f};
00092 #endif
00093
00094 const int kDefaultOrderAccuracy{2};
00095
00096 const int kCriticalOrderAccuracyGrad{10};
00097
00098 const int kCriticalOrderAccuracyDiv{8};
00099 }
00100#endif // End of: MTK_INCLUDE_ROOTS_H_

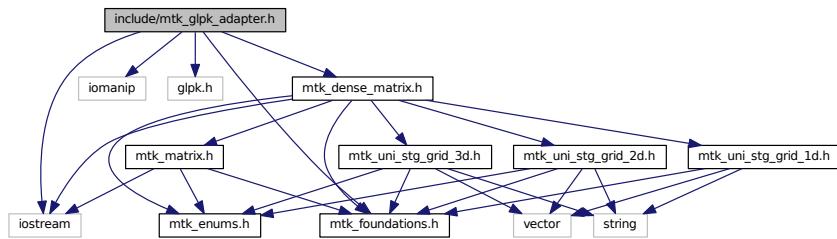
```

18.47 include/mtk_glpk_adapter.h File Reference

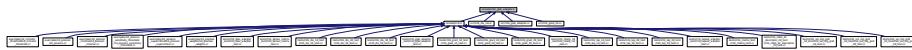
Declaration of an adapter class for the GLPK API.

```
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk_glpk_adapter.h:



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



Classes

- class [mtk::GLPKAdapter](#)
Adapter class for the GLPK API.

Namespaces

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

18.47.1 Detailed Description

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

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

See also

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

Author

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

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

18.48 mtk_glpk_adapter.h

```

00001 /*
00020 Copyright (C) 2016, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00067 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00068
00069 #include <iostream>
00070 #include <iomanip>
00071
00072 #include "glpk.h"
00073
00074 #include "mtk_foundations.h"
00075 #include "mtk_dense_matrix.h"
00076
00077 namespace mtk {
00078
00100 class GLPKAdapter {
00101 public:
00122     static mtk::Real SolveSimplexAndCompare(
        mtk::Real *AA,
00123                                         int nrows,
00124                                         int ncols,
00125                                         int kk,
00126                                         mtk::Real *hh,
00127                                         mtk::Real *qq,
00128                                         int robjective,
00129                                         mtk::Real mimetic_threshold,
00130                                         int copy) noexcept;
00131 };
00132 }
00133 #endif // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_

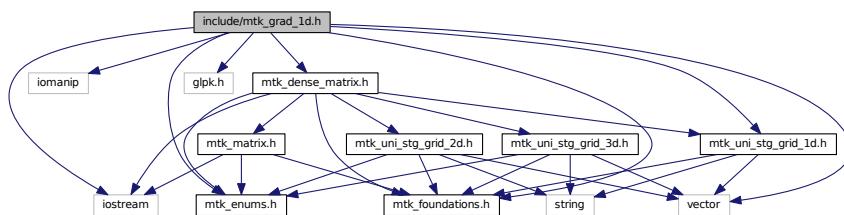
```

18.49 include/mtk_grad_1d.h File Reference

Includes the definition of the class Grad1D.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "glpk.h"
#include "mtk_foundations.h"
#include "mtkEnums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```

Include dependency graph for mtk_grad_1d.h:



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



Classes

- class [mtk::Grad1D](#)
Implements a 1D mimetic gradient operator.

Namespaces

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

18.49.1 Detailed Description

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

Author

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

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

18.50 mtk_grad_1d.h

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_GRAD_1D_H_
00058 #define MTK_INCLUDE_GRAD_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include <vector>
00064
00065 #include "glpk.h"
00066
00067 #include "mtk_foundations.h"
00068 #include "mtkEnums.h"
00069 #include "mtk_dense_matrix.h"
00070 #include "mtk_uni_stg_grid_1d.h"
00071
00072 namespace mtk {
00073
00084 class Grad1D {
00085 public:
00089   friend std::ostream& operator <<(std::ostream& stream, Grad1D &in);
00090
00094   Grad1D();
00095
00101   Grad1D(const Grad1D &grad);
00102
00106   ~Grad1D();
00107
00113   bool ConstructGrad1D(int order_accuracy = kDefaultOrderAccuracy,
00114                         Real mimetic_threshold = kDefaultMimeticThreshold);
00115
00121   int num_bndy_coeffs() const;
```

```

00122
00128     Real *coeffs_interior() const;
00129
00135     Real *weights_crs(void) const;
00136
00142     Real *weights_cbs(void) const;
00143
00149     int num_feasible_sols() const;
00150
00156     DenseMatrix mim_bndy() const;
00157
00163     std::vector<Real> sums_rows_mim_bndy() const;
00164
00170     Real mimetic_measure() const;
00171
00177     DenseMatrix ReturnAsDenseMatrix(Real west,
00178                                         Real east, int num_cells_x) const;
00179
00184     DenseMatrix ReturnAsDenseMatrix(const
00185                                         UniStgGrid1D &grid) const;
00186
00191     DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
00192     const;
00193
00194 private:
00195     bool ComputeStencilInteriorGrid(void);
00196
00197     bool ComputeRationalBasisNullSpace(void);
00198
00199     bool ComputePreliminaryApproximations(void);
00200
00201     bool ComputeWeights(void);
00202
00203     bool ComputeStencilBoundaryGrid(void);
00204
00205     bool AssembleOperator(void);
00206
00207     int order_accuracy_;
00208     int dim_null_;
00209     int num_bndy_approxos_;
00210     int num_bndy_coeffs_;
00211     int gradient_length_;
00212     int minrow_;
00213     int row_;
00214     int num_feasible_sols_;
00215
00216     DenseMatrix rat_basis_null_space_;
00217
00218     Real *coeffs_interior_;
00219     Real *prem_apps_;
00220     Real *weights_crs_;
00221     Real *weights_cbs_;
00222     Real *mim_bndy_;
00223     Real *gradient_;
00224
00225     Real mimetic_threshold_;
00226     Real mimetic_measure_;
00227
00228     std::vector<Real> sums_rows_mim_bndy_;
00229 }
00230 }
00231 #endif // End of: MTK_INCLUDE_GRAD_1D_H_

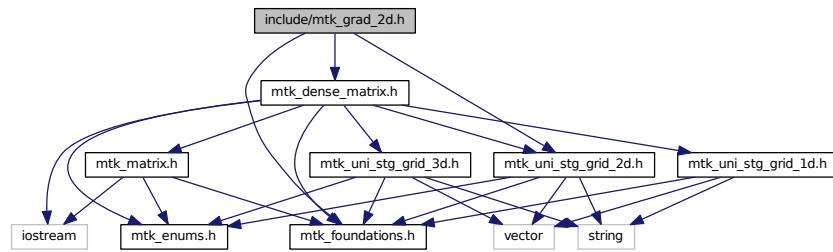
```

18.51 include/mtk_grad_2d.h File Reference

Includes the definition of the class Grad2D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk_grad_2d.h:



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



Classes

- class [mtk::Grad2D](#)
Implements a 2D mimetic gradient operator.

Namespaces

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

18.51.1 Detailed Description

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

Author

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

Definition in file [mtk_grad_2d.h](#).

18.52 mtk_grad_2d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
  
```

```

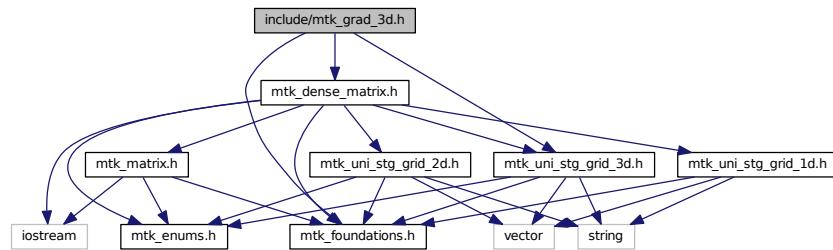
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csdc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_2D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_2D_H_
00059
00060 #include "mtk_foundations.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Grad2D {
00077 public:
00081     Grad2D();
00082
00088     Grad2D(const Grad2D &grad);
00089
00093     ~Grad2D();
00094
00100    bool ConstructGrad2D(const UniStgGrid2D &grid,
00101                      int order_accuracy = kDefaultOrderAccuracy,
00102                      Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109    DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112     DenseMatrix gradient_;
00113
00114     int order_accuracy_;
00115
00116     Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_GRAD_2D_H_

```

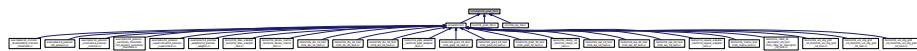
18.53 include/mtk_grad_3d.h File Reference

Includes the definition of the class Grad3D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"
Include dependency graph for mtk_grad_3d.h:
```



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



Classes

- class [mtk::Grad3D](#)
Implements a 3D mimetic gradient operator.

Namespaces

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

18.53.1 Detailed Description

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

Author

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

Definition in file [mtk_grad_3d.h](#).

18.54 mtk_grad_3d.h

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
```

```

00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_3D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_3D_H_
00059
00060 #include "mtk_foundations.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{
00065
00076 class Grad3D {
00077 public:
00081     Grad3D();
00082
00088     Grad3D(const Grad3D &grad);
00089
00093     ~Grad3D();
00094
00100     bool ConstructGrad3D(const UniStgGrid3D &grid,
00101             int order_accuracy = kDefaultOrderAccuracy,
00102             Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109     DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112     DenseMatrix gradient_;
00113
00114     int order_accuracy_;
00115
00116     Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_GRAD_3D_H_

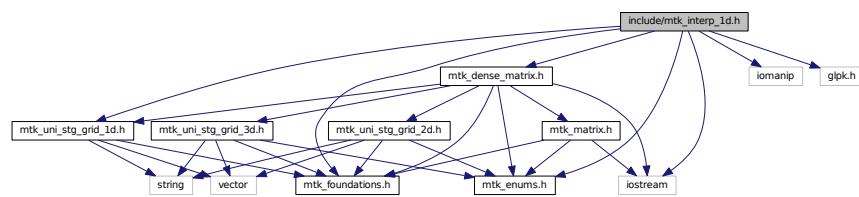
```

18.55 include/mtk_interp_1d.h File Reference

Includes the definition of the class `Interp1D`.

```
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_foundations.h"
#include "mtkEnums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```

Include dependency graph for `mtk_interp_1d.h`:



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



Classes

- class [mtk::Interp1D](#)
Implements a 1D interpolation operator.

Namespaces

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

18.55.1 Detailed Description

Definition of a class that implements a 1D interpolation operator.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez@mail.sdsu.edu
: Johnny Corbino - jcorbino@mail.sdsu.edu

Definition in file [mtk_interp_1d.h](#).

18.56 mtk_interp_1d.h

```

00001
00012 /*
00013 Copyright (C) 2016, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_INTERP_1D_H_
00059 #define MTK_INCLUDE_INTERP_1D_H_
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "glpk.h"
00065
00066 #include "mtk_foundations.h"
00067 #include "mtkEnums.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:
00087   friend std::ostream& operator <<(std::ostream& stream, Interp1D &in);
00088
00092   Interp1D();
00093
00099   Interp1D(const Interp1D &interp);
00100
00104   ~Interp1D();
00105
00111   bool ConstructInterp1D(int order_accuracy =
00112     kDefaultOrderAccuracy,
00113     mtk::DirInterp dir =
00114     mtk::DirInterp::SCALAR_TO_VECTOR);
00115
00119   Real *coeffs_interior() const;

```

```

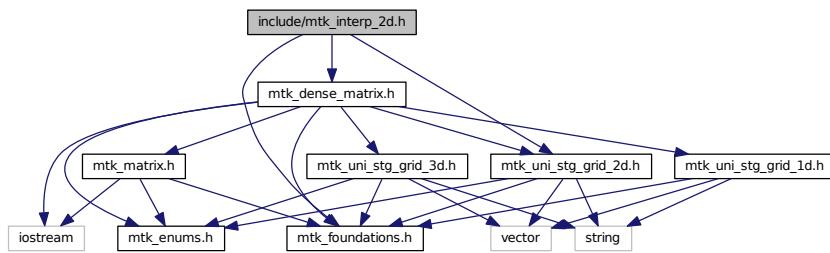
00120     DenseMatrix ReturnAsDenseMatrix(const
00121         UniStgGrid1D &grid) const;
00122
00123     private:
00124     DirInterp dir_interp_;
00125
00126     int order_accuracy_;
00127
00128     Real *coeffs_interior_;
00129
00130 };
00131
00132
00133 #endif // End of: MTK_INCLUDE_INTERP_1D_H_

```

18.57 include/mtk_interp_2d.h File Reference

Includes the definition of the class Interp2D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_interp_2d.h:
```



Classes

- class [mtk::Interp2D](#)
Implements a 2D interpolation operator.

Namespaces

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

18.57.1 Detailed Description

This class implements a 2D interpolation operator.

Author

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

Definition in file [mtk_interp_2d.h](#).

18.58 mtk_interp_2d.h

```

00001
00012 /*
00013 Copyright (C) 2016, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_MTK_INTERP_2D_H_
00059 #define MTK_INCLUDE_MTK_INTERP_2D_H_
00060
00061 #include "mtk_foundations.h"
00062 #include "mtk_dense_matrix.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk{
00066
00076 class Interp2D {
00077 public:
00081     Interp2D();
00082
00088     Interp2D(const Interp2D &interp);
00089
00093     ~Interp2D();
00094
00100     DenseMatrix ConstructInterp2D(const UniStgGrid2D &grid,
00101                                     int order_accuracy = kDefaultOrderAccuracy,
00102                                     Real mimetic_threshold =
00103                                     kDefaultMimeticThreshold);
00109     DenseMatrix ReturnAsDenseMatrix();
00110
00111 private:
00112     DenseMatrix interpolator_;
00113
00114     int order_accuracy_;
00115
00116     Real mimetic_threshold_;
00117 };

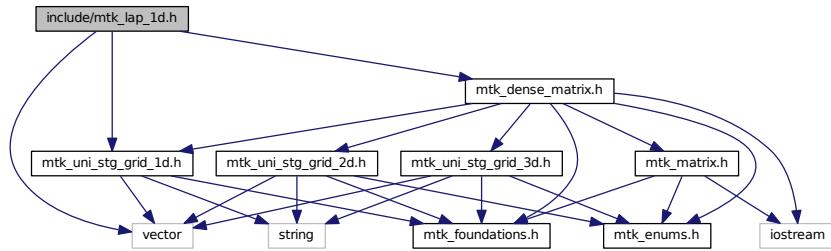
```

```
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_INTERP_2D_H_
```

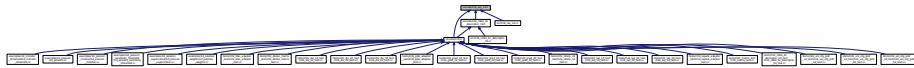
18.59 include/mtk_lap_1d.h File Reference

Includes the definition of the class Lap1D.

```
#include <vector>
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
Include dependency graph for mtk_lap_1d.h:
```



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



Classes

- class [mtk::Lap1D](#)
Implements a 1D mimetic Laplacian operator.

Namespaces

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

18.59.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_1d.h](#).

18.60 mtk_lap_1d.h

```

00001
00011 

```

```

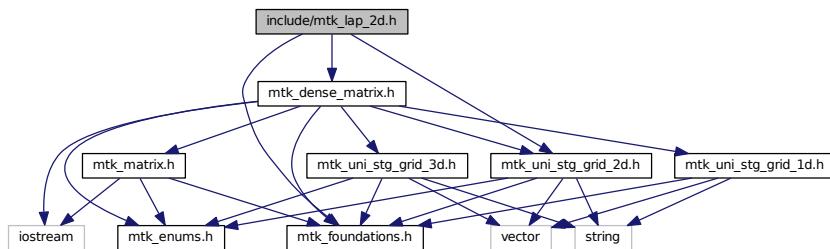
00137     Real mimetic_measure() const;
00143
00144
00150     DenseMatrix ReturnAsDenseMatrix(const
00151         UniStgGrid1D &grid) const;
00157     const mtk::Real* data(const UniStgGrid1D &grid) const;
00158
00159     private:
00160     int order_accuracy_;
00161     int laplacian_length_;
00162
00163     Real *laplacian_;
00164
00165     mutable Real delta_;
00166
00167     Real mimetic_threshold_;
00168     Real mimetic_measure_;
00169
00170     std::vector<Real> sums_rows_mim_bndy_;
00171 };
00172 }
00173 #endif // End of: MTK_INCLUDE_LAP_1D_H_

```

18.61 include/mtk_lap_2d.h File Reference

Includes the implementation of the class Lap2D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_lap_2d.h:
```



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



Classes

- class `mtk::Lap2D`

Implements a 2D mimetic Laplacian operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.61.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_2d.h](#).

18.62 mtk_lap_2d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_2D_H_
00058 #define MTK_INCLUDE_MTK_LAP_2D_H_
00059
00060 #include "mtk_foundations.h"

```

```

00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap2D {
00077 public:
00081     Lap2D();
00082
00088     Lap2D(const Lap2D &lap);
00089
00093     ~Lap2D();
00094
00100     bool ConstructLap2D(const UniStgGrid2D &grid,
00101                     int order_accuracy = kDefaultOrderAccuracy,
00102                     Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109     DenseMatrix ReturnAsDenseMatrix() const;
00110
00116     Real *data() const;
00117
00118 private:
00119     DenseMatrix laplacian_;
00120
00121     int order_accuracy_;
00122
00123     Real mimetic_threshold_;
00124 };
00125 }
00126 #endif // End of: MTK_INCLUDE_MTK_LAP_2D_H_

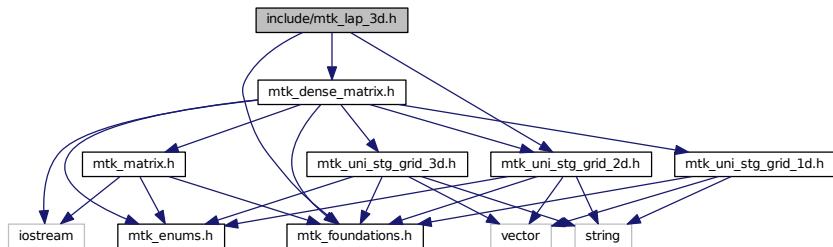
```

18.63 include/mtk_lap_3d.h File Reference

Includes the implementation of the class Lap3D.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk_lap_3d.h:



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



Classes

- class [mtk::Lap3D](#)

Implements a 3D mimetic Laplacian operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.63.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_3d.h](#).

18.64 mtk_lap_3d.h

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */

```

```

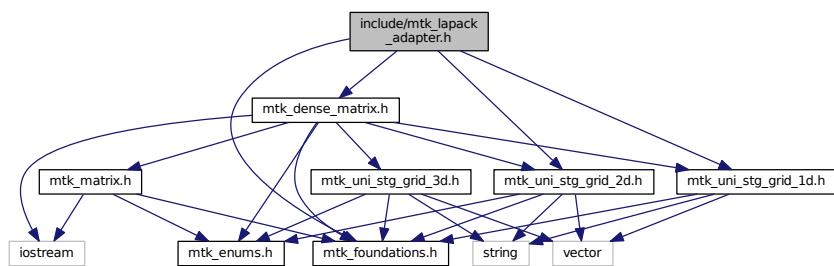
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_3D_H_
00058 #define MTK_INCLUDE_MTK_LAP_3D_H_
00059
00060 #include "mtk_foundations.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap3D {
00077 public:
00081     UniStgGrid3D operator*(const UniStgGrid3D &grid) const;
00082
00086     Lap3D();
00087
00093     Lap3D(const Lap3D &lap);
00094
00098     ~Lap3D();
00099
00105     bool ConstructLap3D(const UniStgGrid3D &grid,
00106                           int order_accuracy = kDefaultOrderAccuracy,
00107                           Real mimetic_threshold = kDefaultMimeticThreshold);
00108
00114     DenseMatrix ReturnAsDenseMatrix() const;
00115
00121     Real *data() const;
00122
00123 private:
00124     DenseMatrix laplacian_;
00125
00126     int order_accuracy_;
00127
00128     Real mimetic_threshold_;
00129 };
00130 }
00131 #endif // End of: MTK_INCLUDE_MTK_LAP_3D_H_

```

18.65 include/mtk_lapack_adapter.h File Reference

Declaration of an adapter class for the LAPACK API.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_lapack_adapter.h:
```



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



Classes

- class [mtk::LAPACKAdapter](#)
Adapter class for the LAPACK API.

Namespaces

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

18.65.1 Detailed Description

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

The **LAPACK (Linear Algebra PACKAGE)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Author

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

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

18.66 mtk_lapack_adapter.h

```

00001
00020 /*
00021 Copyright (C) 2016, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,

```

```

00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00067 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00068
00069 #include "mtk_foundations.h"
00070 #include "mtk_dense_matrix.h"
00071 #include "mtk_uni_stg_grid_1d.h"
00072 #include "mtk_uni_stg_grid_2d.h"
00073
00074 namespace mtk {
00075
00094 class LAPACKAdapter {
00095 public:
00106     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00107                                 mtk::Real *rhs);
00108
00119     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00120                                 mtk::DenseMatrix &rr);
00121
00132     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00133                                 mtk::UniStgGrid1D &rhs);
00134
00135
00146     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00147                                 mtk::UniStgGrid2D &rhs);
00148
00160     static int SolveRectangularDenseSystem(const
00161                                             mtk::DenseMatrix &aa,
00162                                             mtk::Real *ob_,
00163                                             int ob_ld_);
00175     static mtk::DenseMatrix QRFactorDenseMatrix(
00176         DenseMatrix &matrix);
00176 };
00177 }
00178#endif // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_

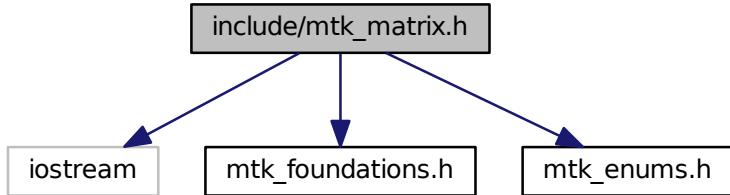
```

18.67 include/mtk_matrix.h File Reference

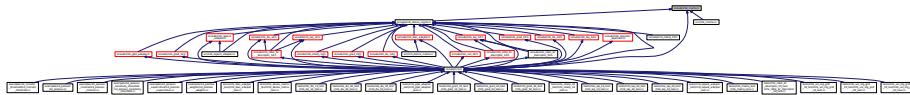
Declaration of the representation of a matrix in the MTK.

```
#include <iostream>
#include "mtk_foundations.h"
#include "mtk_enums.h"
```

Include dependency graph for mtk_matrix.h:



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



Classes

- class [mtk::Matrix](#)
Definition of the representation of a matrix in the MTK.

Namespaces

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

18.67.1 Detailed Description

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

Author

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

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

18.68 mtk_matrix.h

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
  
```

```
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_MATRIX_H_
00057 #define MTK_INCLUDE_MATRIX_H_
00058
00059 #include <iostream>
00060
00061 #include "mtk_foundations.h"
00062 #include "mtkEnums.h"
00063
00064 namespace mtk {
00065
00075 class Matrix {
00076 public:
00080     Matrix();
00081
00087     Matrix(const Matrix &in);
00088
00092     ~Matrix() noexcept;
00093
00101     EncodedOperator encoded_operator() const noexcept;
00102
00110     MatrixStorage storage() const noexcept;
00111
00119     MatrixOrdering ordering() const noexcept;
00120
00126     int num_rows() const noexcept;
00127
00133     int num_cols() const noexcept;
00134
00140     int num_values() const noexcept;
00141
00151     int leading_dimension() const noexcept;
00152
00158     int num_zero() const noexcept;
00159
00165     int num_non_zero() const noexcept;
00166
00174     int num_null() const noexcept;
00175
00183     int num_non_null() const noexcept;
```

```
00184
00185     int num_low_diags() const noexcept;
00186
00187     int num_upp_diags() const noexcept;
00188
00189     int bandwidth() const noexcept;
00190
00191     Real abs_density() const noexcept;
00192
00193     Real rel_density() const noexcept;
00194
00195     Real abs_sparsity() const noexcept;
00196
00197     Real rel_sparsity() const noexcept;
00198
00199 void set_encoded_operator(const EncodedOperator &in) noexcept;
00200
00201 void set_storage(const MatrixStorage &tt) noexcept;
00202
00203 void set_ordering(const MatrixOrdering &oo) noexcept;
00204
00205 void set_num_rows(const int &num_rows) noexcept;
00206
00207 void set_num_cols(const int &num_cols) noexcept;
00208
00209 void set_num_low_diags(const int &num_low_diags) noexcept;
00210
00211 void set_num_upp_diags(const int &num_upp_diags) noexcept;
00212
00213 void set_num_null(const int &num_null) noexcept;
00214
00215 void set_num_zero(const int &num_zero) noexcept;
00216
00217 void IncreaseNumNull() noexcept;
00218
00219 void DecreaseNumNull() noexcept;
00220
00221 void IncreaseNumZero() noexcept;
00222
00223 void DecreaseNumZero() noexcept;
00224
00225 private:
00226     int ComputeNumValues(const int &num_rows, const int &num_cols) const noexcept;
00227
00228     int ComputeLeadingDimension(const int &num_rows, const int &num_cols)
00229         const noexcept;
00230
00231     int ComputeBandwidth(const int &num_low_diags, const int &num_upp_diags)
00232         const noexcept;
00233
00234     int ComputeNumNonNull(const int &num_values, const int &num_null)
00235         const noexcept;
00236
00237     mtk::Real ComputeAbsDensity(const int &num_non_null, const int &
00238         num_values)
00239         const noexcept;
00240
00241     mtk::Real ComputeAbsSparsity(const mtk::Real &absolute_density)
00242         const noexcept;
00243
00244     int ComputeNumNonZero(const int &num_values, const int &num_zero)
00245         const noexcept;
00246
00247     mtk::Real ComputeRelDensity(const int &num_non_zero, const int &
00248         num_values)
00249         const noexcept;
00250
00251     mtk::Real ComputeRelSparsity(const mtk::Real &relative_density)
00252         const noexcept;
00253
00254     EncodedOperator encoded_operator_;
00255
00256     MatrixStorage storage_;
00257
00258     MatrixOrdering ordering_;
00259
00260     int num_rows_;
00261     int num_cols_;
00262     int num_values_;
00263     int leading_dimension_;
```

```

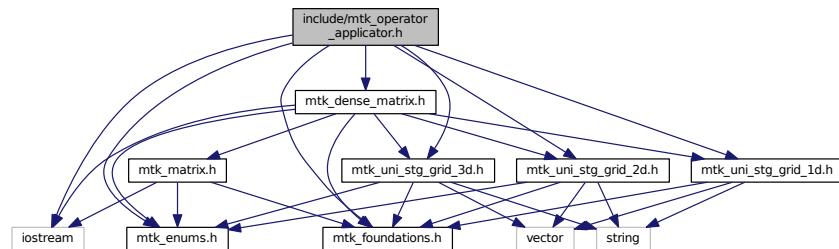
00459 int num_low_diags_;
00460 int num_upp_diags_;
00461 int bandwidth_;
00462
00463 int num_null_;
00464 int num_non_null_;
00465
00466 Real abs_density_;
00467 Real abs_sparsity_;
00468
00469 int num_zero_;
00470 int num_non_zero_;
00471
00472 Real rel_density_;
00473 Real rel_sparsity_;
00474 };
00475 }
00476 #endif // End of: MTK_INCLUDE_MATRIX_H_

```

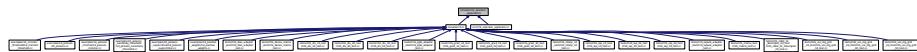
18.69 include/mtk_operator_applicator.h File Reference

Controls the process of applying a mimetic operator to a grid.

```
#include <iostream>
#include "mtk_foundations.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"
Include dependency graph for mtk_operator_applicator.h:
```



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



Classes

- class [mtk::OperatorApplicator](#)

Controls the process of applying a mimetic operator to a grid.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.69.1 Detailed Description

Author

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

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

18.70 mtk_operator_applicator.h

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_OPERATOR_APPPLICATOR_H_
00057 #define MTK_INCLUDE_OPERATOR_APPPLICATOR_H_
00058
00059 #include <iostream>
00060
00061 #include "mtk_foundations.h"
00062 #include "mtkEnums.h"
00063 #include "mtkDenseMatrix.h"
```

```

00064 #include "mtk_uni_stg_grid_1d.h"
00065 #include "mtk_uni_stg_grid_2d.h"
00066 #include "mtk_uni_stg_grid_3d.h"
00067
00068 namespace mtk {
00069
00079 class OperatorApplicator {
00080 public:
00088     static void ApplyDenseMatrixGradientOn1DGrid(
00089         DenseMatrix &grad,
00090                               UniStgGrid1D &grid,
00091                               UniStgGrid1D &out);
00099     static void ApplyDenseMatrixDivergenceOn1DGrid(
00100         DenseMatrix &div,
00101                               UniStgGrid1D &grid,
00102                               UniStgGrid1D &out);
00110     static void ApplyDenseMatrixLaplacianOn1DGrid(
00111         DenseMatrix &lap,
00112                               UniStgGrid1D &grid,
00113                               UniStgGrid1D &out);
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_OPERATOR_APPLICATOR_H_

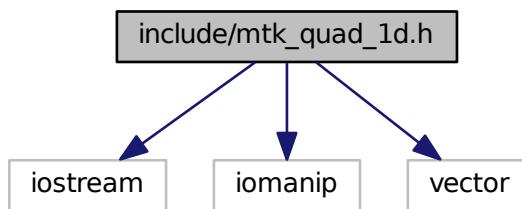
```

18.71 include/mtk_quad_1d.h File Reference

Includes the definition of the class Quad1D.

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

Include dependency graph for mtk_quad_1d.h:



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



Classes

- class [mtk::Quad1D](#)

Implements a 1D mimetic quadrature.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.71.1 Detailed Description

Definition of a class that implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

See also

[mtk::Grad1D](#)

Author

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

Todo Implement this class.

Definition in file [mtk_quad_1d.h](#).

18.72 mtk_quad_1d.h

```

00001
00015 /*
00016 Copyright (C) 2016, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON

```

```

00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_QUAD_1D_H_
00062 #define MTK_INCLUDE_QUAD_1D_H_
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #include <vector>
00068
00069 namespace mtk {
00070
00081 class Quad1D {
00082 public:
00086   friend std::ostream& operator <<(std::ostream& stream, Quad1D &in);
00087
00091   Quad1D();
00092
00098   Quad1D(const Quad1D &quad);
00099
00103   ~Quad1D();
00104
00110   int degree_approximation() const;
00111
00117   Real *weights() const;
00118
00127   Real Integrate(Real (*Integrand)(Real xx), UniStgGrid1D grid) const;
00128
00129 private:
00130   int degree_approximation_;
00131
00132   std::vector<Real> weights_;
00133 };
00134 }
00135 #endif // End of: MTK_INCLUDE_QUAD_1D_H_

```

18.73 include/mtk_robin_bc_descriptor_1d.h File Reference

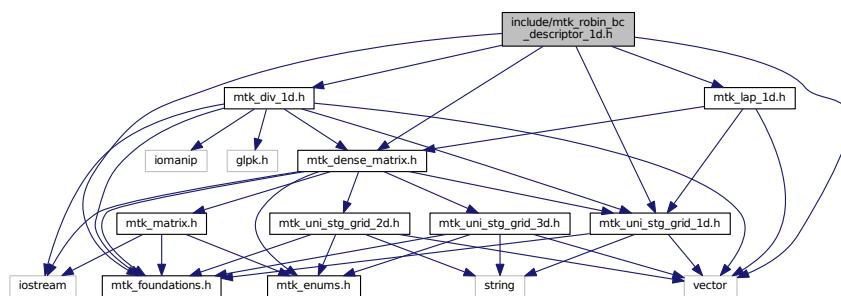
Impose Robin boundary conditions on the operators and on the grids.

```

#include <vector>
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"

```

Include dependency graph for mtk_robin_bc_descriptor_1d.h:



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



Classes

- class [mtk::RobinBCDescriptor1D](#)

Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

TypeDefs

- `typedef Real(* mtk::CoefficientFunction0D)(const Real &t, const std::vector< Real > &p)`
A function of a BC coefficient evaluated on a 0D domain and time.

18.73.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\begin{aligned} \delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) &= \beta_a(a, t), \\ \delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) &= \beta_b(b, t). \end{aligned}$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

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

Definition in file [mtk_robin_bc_descriptor_1d.h](#).

18.74 mtk_robin_bc_descriptor_1d.h

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <vector>
00090
00091 #include "mtk_foundations.h"
00092 #include "mtk_dense_matrix.h"
00093 #include "mtk_uni_stg_grid_1d.h"
00094 #include "mtk_div_1d.h"
00095 #include "mtk_lap_1d.h"
00096
00097 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00098 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00099
00100 namespace mtk {
00108     typedef Real (*CoefficientFunction0D)(const Real &tt,
00109                                         const std::vector<Real> &pp);
00110
00153     class RobinBCDescriptor1D {
00154     public:
00155         RobinBCDescriptor1D();
00156         RobinBCDescriptor1D(const RobinBCDescriptor1D &desc);
00157         ~RobinBCDescriptor1D() noexcept;
00158         int highest_order_diff_west() const noexcept;
00159         int highest_order_diff_east() const noexcept;
00160         void PushBackWestCoeff(CoefficientFunction0D cw);
00161         void PushBackEastCoeff(CoefficientFunction0D ce);
00162     };
00163 }
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199

```

```

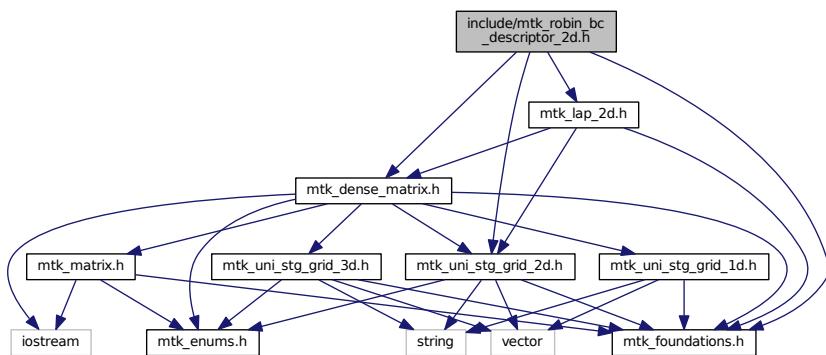
00205 void set_west_condition(Real (*west_condition)(const
00206 Real &tt)) noexcept;
00212 void set_east_condition(Real (*east_condition)(const
00213 Real &tt)) noexcept;
00223 bool ImposeOnDivergenceMatrix(
00224 const Div1D &div,
00225 DenseMatrix &matrix,
00226 const std::vector<Real> &parameters = std::vector<Real>(),
00227 const Real &time = mtk::kZero) const;
00228
00238 bool ImposeOnLaplacianMatrix(
00239 const Lap1D &lap,
00240 DenseMatrix &matrix,
00241 const std::vector<Real> &parameters = std::vector<Real>(),
00242 const Real &time = mtk::kZero) const;
00249 void ImposeOnGrid(UniStgGrid1D &grid, const Real &time =
00250 mtk::kZero) const;
00251 private:
00252 int highest_order_diff_west_;
00253 int highest_order_diff_east_;
00254
00255 std::vector<CoefficientFunction0D> west_coefficients_;
00256 std::vector<CoefficientFunction0D> east_coefficients_;
00257
00258 Real (*west_condition_)(const Real &tt);
00259 Real (*east_condition_)(const Real &tt);
00260 };
00261 }
00262 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_

```

18.75 include/mtk_robin_bc_descriptor_2d.h File Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_robin_bc_descriptor_2d.h:
```



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



Classes

- class [mtk::RobinBCDescriptor2D](#)

Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

TypeDefs

- [typedef Real\(* mtk::CoefficientFunction1D \)](#)(const Real &xx, const Real &tt)

A function of a BC coefficient evaluated on a 1D domain and time.

18.75.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

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

Definition in file [mtk_robin_bc_descriptor_2d.h](#).

18.76 mtk_robin_bc_descriptor_2d.h

```

00001
00034 

```

```

00193 void PushBackEastCoeff(CoefficientFunction1D ce);
00194
00201 void PushBackSouthCoeff(CoefficientFunction1D cs);
00202
00209 void PushBackNorthCoeff(CoefficientFunction1D cn);
00210
00217 void set_west_condition(Real (*west_condition)(const
00218 Real &yy,
00219                                     const Real &tt)) noexcept;
00226 void set_east_condition(Real (*east_condition)(const
00227 Real &yy,
00228                                     const Real &tt)) noexcept;
00235 void set_south_condition(Real (*south_condition)(const
00236 Real &xx,
00237                                     const Real &tt)) noexcept;
00244 void set_north_condition(Real (*north_condition)(const
00245 Real &xx,
00246                                     const Real &tt)) noexcept;
00255 bool ImposeOnLaplacianMatrix(const Lap2D &lap,
00256                                     const UniStgGrid2D &grid,
00257                                     DenseMatrix &matrix,
00258                                     const Real &time = kZero) const;
00265 void ImposeOnGrid(UniStgGrid2D &grid, const Real &time =
00266 kZero) const;
00267 private:
00276 bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00277                                     const UniStgGrid2D &grid,
00278                                     DenseMatrix &matrix,
00279                                     const Real &time = kZero) const;
00288 bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00289                                     const UniStgGrid2D &grid,
00290                                     DenseMatrix &matrix,
00291                                     const Real &time = kZero) const;
00300 bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00301                                     const UniStgGrid2D &grid,
00302                                     DenseMatrix &matrix,
00303                                     const Real &time = kZero) const;
00312 bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00313                                     const UniStgGrid2D &grid,
00314                                     DenseMatrix &matrix,
00315                                     const Real &time = kZero) const;
00324 bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00325                                     const UniStgGrid2D &grid,
00326                                     DenseMatrix &matrix,
00327                                     const Real &time = kZero) const;
00336 bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00337                                     const UniStgGrid2D &grid,
00338                                     DenseMatrix &matrix,
00339                                     const Real &time = kZero) const;
00348 bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00349                                     const UniStgGrid2D &grid,
00350                                     DenseMatrix &matrix,
00351                                     const Real &time = kZero) const;
00360 bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00361                                     const UniStgGrid2D &grid,
00362                                     DenseMatrix &matrix,
00363                                     const Real &time = kZero) const;
00364
00365 int highest_order_diff_west_;
00366 int highest_order_diff_east_;
00367 int highest_order_diff_south_;
00368 int highest_order_diff_north_;
00369
00370 std::vector<CoefficientFunction1D> west_coefficients_;
00371 std::vector<CoefficientFunction1D> east_coefficients_;
00372 std::vector<CoefficientFunction1D> south_coefficients_;
00373 std::vector<CoefficientFunction1D> north_coefficients_;
00374
00375 Real (*west_condition_)(const Real &xx, const Real &tt);
00376 Real (*east_condition_)(const Real &xx, const Real &tt);
00377 Real (*south_condition_)(const Real &yy, const Real &tt);
00378 Real (*north_condition_)(const Real &yy, const Real &tt);
00379 };
00380 }
00381 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_

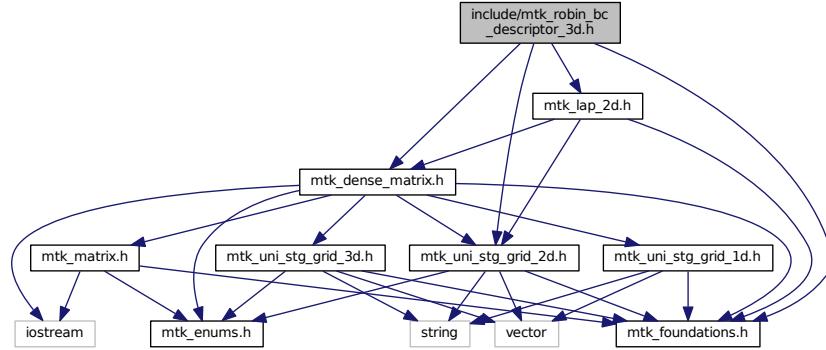
```

18.77 include/mtk_robin_bc_descriptor_3d.h File Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include "mtk_foundations.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"
```

Include dependency graph for mtk_robin_bc_descriptor_3d.h:



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



Classes

- class [mtk::RobinBCDescriptor3D](#)

Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Typedefs

- `typedef Real(* mtk::CoefficientFunction2D)(const Real &xx, const Real &yy, const Real &tt)`

A function of a BC coefficient evaluated on a 2D domain and time.

18.77.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

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

Definition in file [mtk_robin_bc_descriptor_3d.h](#).

18.78 mtk_robin_bc_descriptor_3d.h

```

00001
00034 /*
00035 Copyright (C) 2016, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00049
00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067

```

```

00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00081 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00082
00083 #include "mtk_foundations.h"
00084 #include "mtk_dense_matrix.h"
00085 #include "mtk_lap_2d.h"
00086 #include "mtk_uni_stg_grid_2d.h"
00087
00088 namespace mtk{
00089
00097 typedef Real (*CoefficientFunction2D)(const Real &xx,
00098                                     const Real &yy,
00099                                     const Real &tt);
00100
00134 class RobinBCDescriptor3D {
00135 public:
00139     RobinBCDescriptor3D();
00140
00146     RobinBCDescriptor3D(const RobinBCDescriptor3D &desc);
00147
00151     ~RobinBCDescriptor3D() noexcept;
00152
00158     int highest_order_diff_west() const noexcept;
00159
00166     void PushBackWestCoeff(CoefficientFunction2D cw);
00167
00174     void set_west_condition(Real (*west_condition)(const
00175                                         Real &xx,
00176                                         const Real &yy,
00177                                         const Real &tt)) noexcept;
00178
00186     bool ImposeOnLaplacianMatrix(const Lap3D &lap,
00187                                   const UniStgGrid3D &grid,
00188                                   DenseMatrix &matrix,
00189                                   const Real &time = kZero) const;
00196     void ImposeOnGrid(UniStgGrid3D &grid, const Real &time =
00197                         kZero) const;
00198 private:
00207     bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00208                                       const UniStgGrid2D &grid,
00209                                       DenseMatrix &matrix,
00210                                       const Real &time = kZero) const;
00219     bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00220                                       const UniStgGrid2D &grid,
00221                                       DenseMatrix &matrix,
00222                                       const Real &time = kZero) const;
00231     bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00232                                       const UniStgGrid2D &grid,
00233                                       DenseMatrix &matrix,
00234                                       const Real &time = kZero) const;
00243     bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00244                                       const UniStgGrid2D &grid,
00245                                       DenseMatrix &matrix,
00246                                       const Real &time = kZero) const;
00255     bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00256                                         const UniStgGrid2D &grid,
00257                                         DenseMatrix &matrix,
00258                                         const Real &time = kZero) const;
00267     bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00268                                         const UniStgGrid2D &grid,
00269                                         DenseMatrix &matrix,
00270                                         const Real &time = kZero) const;
00279     bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00280                                         const UniStgGrid2D &grid,
00281                                         DenseMatrix &matrix,
00282                                         const Real &time = kZero) const;
00291     bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00292                                         const UniStgGrid2D &grid,

```

```

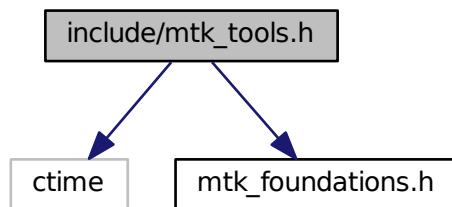
00293                               DenseMatrix &matrix,
00294                               const Real &time = kZero) const;
00295
00296     int highest_order_diff_west_;
00297     int highest_order_diff_east_;
00298     int highest_order_diff_south_;
00299     int highest_order_diff_north_;
00300     int highest_order_diff_bottom_;
00301     int highest_order_diff_top_;
00302
00303     std::vector<CoefficientFunction2D> west_coefficients_;
00304     std::vector<CoefficientFunction2D> east_coefficients_;
00305     std::vector<CoefficientFunction2D> south_coefficients_;
00306     std::vector<CoefficientFunction2D> north_coefficients_;
00307     std::vector<CoefficientFunction2D> bottom_coefficients_;
00308     std::vector<CoefficientFunction2D> top_coefficients_;
00309
00310     Real (*west_condition_)(const Real &xx,
00311                             const Real &yy,
00312                             const Real &tt);
00313     Real (*east_condition_)(const Real &xx,
00314                             const Real &yy,
00315                             const Real &tt);
00316     Real (*south_condition_)(const Real &xx,
00317                             const Real &yy,
00318                             const Real &tt);
00319     Real (*north_condition_)(const Real &xx,
00320                             const Real &yy,
00321                             const Real &tt);
00322     Real (*bottom_condition_)(const Real &xx,
00323                             const Real &yy,
00324                             const Real &tt);
00325     Real (*top_condition_)(const Real &xx,
00326                             const Real &yy,
00327                             const Real &tt);
00328 };
00329 }
00330 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_

```

18.79 include/mtk_tools.h File Reference

Declaration of a class to manage run-time tools.

```
#include <ctime>
#include "mtk_foundations.h"
Include dependency graph for mtk_tools.h:
```



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



Classes

- class [mtk::Tools](#)

Tool manager class.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.79.1 Detailed Description

Declaration of a class providing basic tools to ensure execution correctness, and to assist with unit testing.

Author

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

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

18.80 mtk_tools.h

```

00001 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that

```

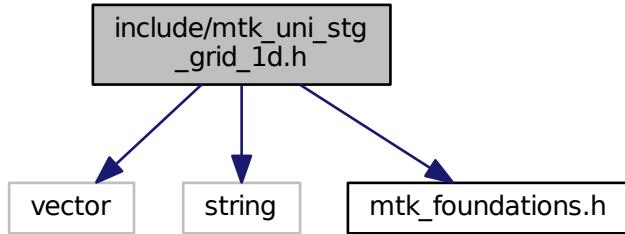
```
00043 parties intellectual property rights.  
00044  
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND  
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED  
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR  
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;  
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON  
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS  
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
00055 */  
00056  
00057 #ifndef MTK_INCLUDE_TOOLS_H_  
00058 #define MTK_INCLUDE_TOOLS_H_  
00059  
00060 #include <ctime>  
00061  
00062 #include "mtk_foundations.h"  
00063  
00064 namespace mtk {  
00065  
00076 class Tools {  
00077 public:  
00088     static void Prevent(const bool complement,  
00089                           const char *const fname,  
00090                           int lineno,  
00091                           const char *const fxname) noexcept;  
00092  
00098     static void BeginUnitTestNo(const int &nn) noexcept;  
00099  
00105     static void EndUnitTestNo(const int &nn) noexcept;  
00106  
00112     static void Assert(const bool &condition) noexcept;  
00113  
00114 private:  
00115     static int test_number_;  
00116  
00117     static Real duration_;  
00118  
00119     static clock_t begin_time_;  
00120 };  
00121 }  
00122 #endif // End of: MTK_INCLUDE_TOOLS_H_
```

18.81 include/mtk_uni_stg_grid_1d.h File Reference

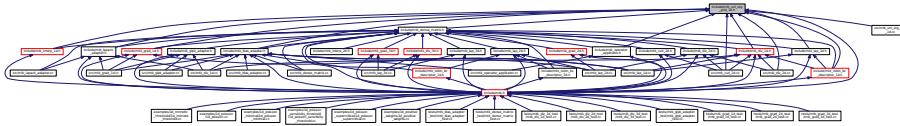
Definition of an 1D uniform staggered grid.

```
#include <vector>  
#include <string>  
#include "mtk_foundations.h"
```

Include dependency graph for mtk_uni_stg_grid_1d.h:



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



Classes

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

Namespaces

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

18.81.1 Detailed Description

Definition of an 1D uniform staggered grid.

Author

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

Todo Create overloaded binding routines that read data from files.

Definition in file [mtk_uni_stg_grid_1d.h](#).

18.82 mtk_uni_stg_grid_1d.h

```

00001
00012 /*
00013 Copyright (C) 2016, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_foundations.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078 public:
00082     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00083
00091     UniStgGrid1D& operator =(const UniStgGrid1D &in);
00092
00096     UniStgGrid1D();
00097
00103     UniStgGrid1D(const UniStgGrid1D &grid);
00104
00115     UniStgGrid1D(const Real &west_bndy_x,
00116                  const Real &east_bndy_x,
00117                  const int &num_cells_x,
00118                  const mtk::FieldNature &field_nature =
00119                  mtk::FieldNature::SCALAR);
00123     ~UniStgGrid1D();
00124
00130     Real west_bndy_x() const;
00131
00137     Real east_bndy_x() const;

```

```

00138
00144     Real delta_x() const;
00145
00153     const Real *discrete_domain_x() const;
00154
00162     Real *discrete_field();
00163
00169     int num_cells_x() const;
00170
00176     FieldNature field_nature() const;
00177
00181     void GenerateDiscreteDomainX();
00182
00186     void ReserveDiscreteField();
00187
00194     void BindScalarField(
00195         Real (*ScalarField)(const Real &xx, const std::vector<Real> &pp),
00196         const std::vector<Real> &parameters = std::vector<Real>());
00197
00203     void BindScalarField(const std::vector<Real> &samples);
00204
00216     void BindVectorField(
00217         Real (*VectorField)(const Real &xx, const std::vector<Real> &pp),
00218         const std::vector<Real> &parameters = std::vector<Real>());
00219
00231     bool WriteToFile(std::string filename,
00232                     std::string space_name,
00233                     std::string field_name) const;
00234
00235 private:
00236     FieldNature field_nature_;
00237
00238     std::vector<Real> discrete_domain_x_;
00239     std::vector<Real> discrete_field_;
00240
00241     Real west_bndy_x_;
00242     Real east_bndy_x_;
00243     Real num_cells_x_;
00244     Real delta_x_;
00245 };
00246 }
00247 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_

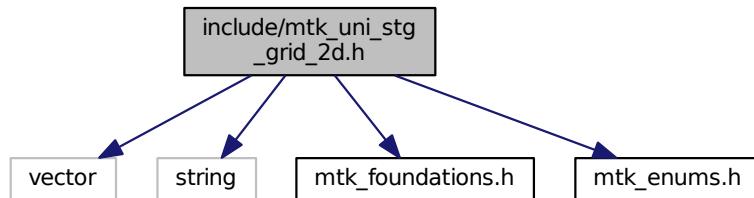
```

18.83 include/mtk_uni_stg_grid_2d.h File Reference

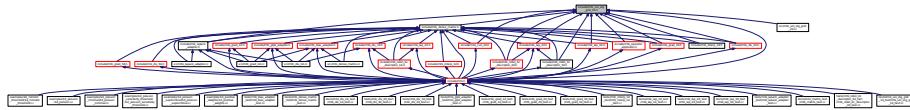
Definition of an 2D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_foundations.h"
#include "mtk_enums.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.

18.83.1 Detailed Description

Definition of an 2D 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.

Todo Create overloaded binding routines that read data from arrays.

Definition in file [mtk_uni_stg_grid_2d.h](#).

18.84 mtk_uni_stg_grid_2d.h

```

00001
00014 /*
00015 Copyright (C) 2016, Computational Science Research Center, San Diego State
00016 University. All rights reserved.
00017
00018 Redistribution and use in source and binary forms, with or without modification,
00019 are permitted provided that the following conditions are met:
00020
00021 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00022 and a copy of the modified files should be reported once modifications are
00023 completed, unless these modifications are made through the project's GitHub
00024 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00025 should be developed and included in any deliverable.
00026
00027 2. Redistributions of source code must be done through direct
00028 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00029
00030 3. 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 4. Usage of the binary form on proprietary applications shall require explicit
00035 prior written permission from the the copyright holders, and due credit should

```

```
00036 be given to the copyright holders.
00037
00038 5. Neither the name of the copyright holder nor the names of its contributors
00039 may be used to endorse or promote products derived from this software without
00040 specific prior written permission.
00041
00042 The copyright holders provide no reassurances that the source code provided does
00043 not infringe any patent, copyright, or any other intellectual property rights of
00044 third parties. The copyright holders disclaim any liability to any recipient for
00045 claims brought against recipient by any third party for infringement of that
00046 parties intellectual property rights.
00047
00048 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00049 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00050 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00051 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00052 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00053 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00054 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00055 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00056 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00057 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00058 */
00059
00060 #ifndef MTK_INCLUDE_UNI_STG_GRID_2D_H_
00061 #define MTK_INCLUDE_UNI_STG_GRID_2D_H_
00062
00063 #include <vector>
00064 #include <string>
00065
00066 #include "mtk_foundations.h"
00067 #include "mtk_enums.h"
00068
00069 namespace mtk {
00070
00080 class UniStgGrid2D {
00081 public:
00085   friend std::ostream& operator <<(std::ostream& stream, UniStgGrid2D &in);
00086
00090   UniStgGrid2D();
00091
00097   UniStgGrid2D(const UniStgGrid2D &grid);
00098
00112   UniStgGrid2D(const Real &west_bndy_x,
00113                 const Real &east_bndy_x,
00114                 const int &num_cells_x,
00115                 const Real &south_bndy_y,
00116                 const Real &north_bndy_y,
00117                 const int &num_cells_y,
00118                 const mtk::FieldNature &nature =
00119                 mtk::FieldNature::SCALAR);
00123   ~UniStgGrid2D();
00124
00132   const Real *discrete_domain_x() const;
00133
00141   const Real *discrete_domain_y() const;
00142
00148   Real *discrete_field();
00149
00157   FieldNature nature() const;
00158
00164   Real west_bndy() const;
00165
00171   Real east_bndy() const;
00172
00178   int num_cells_x() const;
00179
00185   Real delta_x() const;
00186
00192   Real south_bndy() const;
00193
00199   Real north_bndy() const;
00200
00206   int num_cells_y() const;
00207
00213   Real delta_y() const;
00214
00220   bool Bound() const;
00221
00227   int Size() const;
```

```

00228
00229     void BindScalarField(Real (*ScalarField) (const
00230         Real &xx, const Real &yy));
00231
00232     void BindVectorField(Real (*VectorFieldPComponent) (const
00233         Real &xx,
00234                         const Real &yy),
00235         Real (*VectorFieldQComponent) (const
00236         Real &xx,
00237                         const Real &yy));
00238
00239     bool WriteToFile(std::string filename,
00240                      std::string space_name_x,
00241                      std::string space_name_y,
00242                      std::string field_name) const;
00243
00244 private:
00245     void BindVectorFieldPComponent(
00246         Real (*VectorFieldPComponent) (const Real &xx, const
00247             Real &yy));
00248     void BindVectorFieldQComponent (
00249         Real (*VectorFieldQComponent) (const Real &xx, const
00250             Real &yy));
00251
00252     std::vector<Real> discrete_domain_x_;
00253     std::vector<Real> discrete_domain_y_;
00254     std::vector<Real> discrete_field_;
00255
00256     FieldNature nature_;
00257
00258     Real west_bndy_;
00259     Real east_bndy_;
00260     int num_cells_x_;
00261     Real delta_x_;
00262
00263     Real south_bndy_;
00264     Real north_bndy_;
00265     int num_cells_y_;
00266     Real delta_y_;
00267
00268 };
00269
00270 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_

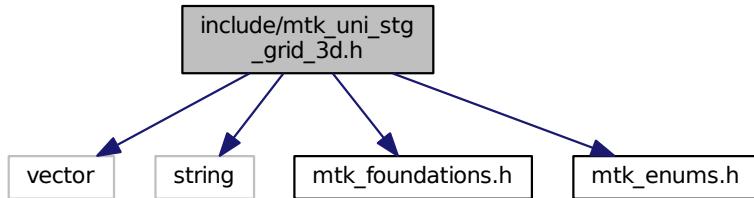
```

18.85 include/mtk_uni_stg_grid_3d.h File Reference

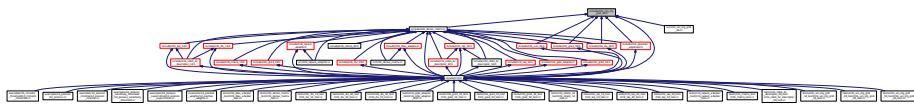
Definition of an 3D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_foundations.h"
#include "mtkEnums.h"

Include dependency graph for mtk_uni_stg_grid_3d.h:
```



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



Classes

- class [mtk::UniStgGrid3D](#)
Uniform 3D Staggered Grid.

Namespaces

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

18.85.1 Detailed Description

Definition of an 3D uniform staggered grid.

Author

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

Todo Create overloaded binding routines that read data from files.

Todo Create overloaded binding routines that read data from arrays.

Definition in file [mtk_uni_stg_grid_3d.h](#).

18.86 mtk_uni_stg_grid_3d.h

```

00001
00014 /*
00015 Copyright (C) 2016, Computational Science Research Center, San Diego State
00016 University. All rights reserved.
00017
00018 Redistribution and use in source and binary forms, with or without modification,
00019 are permitted provided that the following conditions are met:
00020
00021 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00022 and a copy of the modified files should be reported once modifications are
00023 completed, unless these modifications are made through the project's GitHub
00024 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00025 should be developed and included in any deliverable.
00026
00027 2. Redistributions of source code must be done through direct
00028 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00029
00030 3. 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 4. Usage of the binary form on proprietary applications shall require explicit
00035 prior written permission from the the copyright holders, and due credit should

```

```
00036 be given to the copyright holders.
00037
00038 5. Neither the name of the copyright holder nor the names of its contributors
00039 may be used to endorse or promote products derived from this software without
00040 specific prior written permission.
00041
00042 The copyright holders provide no reassurances that the source code provided does
00043 not infringe any patent, copyright, or any other intellectual property rights of
00044 third parties. The copyright holders disclaim any liability to any recipient for
00045 claims brought against recipient by any third party for infringement of that
00046 parties intellectual property rights.
00047
00048 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00049 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00050 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00051 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00052 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00053 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00054 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00055 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00056 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00057 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00058 */
00059
00060 #ifndef MTK_INCLUDE_UNI_STG_GRID_3D_H_
00061 #define MTK_INCLUDE_UNI_STG_GRID_3D_H_
00062
00063 #include <vector>
00064 #include <string>
00065
00066 #include "mtk_foundations.h"
00067 #include "mtk_enums.h"
00068
00069 namespace mtk {
00070
00080 class UniStgGrid3D {
00081 public:
00085   friend std::ostream& operator <<(std::ostream& stream, UniStgGrid3D &in);
00086
00094   UniStgGrid3D operator=(const UniStgGrid3D &in);
00095
00099   UniStgGrid3D();
00100
00106   UniStgGrid3D(const UniStgGrid3D &grid);
00107
00124   UniStgGrid3D(const Real &west_bndy_x,
00125                 const Real &east_bndy_x,
00126                 const int &num_cells_x,
00127                 const Real &south_bndy_y,
00128                 const Real &north_bndy_y,
00129                 const int &num_cells_y,
00130                 const Real &bottom_bndy_z,
00131                 const Real &top_bndy_z,
00132                 const int &num_cells_z,
00133                 const mtk::FieldNature &nature =
00134                 mtk::FieldNature::SCALAR);
00138
00139   ~UniStgGrid3D();
00147
00148   const Real *discrete_domain_x() const;
00156
00157   const Real *discrete_domain_y() const;
00165
00166   const Real *discrete_domain_z() const;
00172
00173   Real *discrete_field();
00181
00182   FieldNature nature() const;
00188
00189   Real west_bndy() const;
00195
00196   Real east_bndy() const;
00202
00203   int num_cells_x() const;
00209
00210   Real delta_x() const;
00216
00217   Real south_bndy() const;
00223
00224   Real north_bndy() const;
```

```

00230     int num_cells_y() const;
00231
00237     Real delta_y() const;
00238
00244     Real bottom_bndy() const;
00245
00251     Real top_bndy() const;
00252
00258     int num_cells_z() const;
00259
00265     Real delta_z() const;
00266
00272     bool Bound() const;
00273
00279     int Size() const;
00280
00286     void BindScalarField(
00287         Real (*ScalarField)(const Real &xx, const Real &yy, const
00288         Real &zz));
00288
00305     void BindVectorField(Real (*VectorFieldPComponent)(const
00306         Real &xx,
00307                         const Real &yy,
00308                         const Real &zz),
00309         Real (*VectorFieldQComponent)(const
00310             Real &xx,
00311                         const Real &yy,
00312                         const Real &zz),
00313                         const Real &zz));
00314
00328     bool WriteToFile(std::string filename,
00329                     std::string space_name_x,
00330                     std::string space_name_y,
00331                     std::string space_name_z,
00332                     std::string field_name) const;
00333
00334 private:
00347     void BindVectorFieldPComponent(
00348         Real (*VectorFieldPComponent)(const Real &xx,
00349                                     const Real &yy,
00350                                     const Real &zz));
00351
00364     void BindVectorFieldQComponent(
00365         Real (*VectorFieldQComponent)(const Real &xx,
00366                                     const Real &yy,
00367                                     const Real &zz));
00368
00381     void BindVectorFieldRComponent(
00382         Real (*VectorFieldRComponent)(const Real &xx,
00383                                     const Real &yy,
00384                                     const Real &zz));
00385
00386     std::vector<Real> discrete_domain_x_;
00387     std::vector<Real> discrete_domain_y_;
00388     std::vector<Real> discrete_domain_z_;
00389     std::vector<Real> discrete_field_;
00390
00391     FieldNature nature_;
00392
00393     Real west_bndy_;
00394     Real east_bndy_;
00395     int num_cells_x_;
00396     Real delta_x_;
00397
00398     Real south_bndy_;
00399     Real north_bndy_;
00400     int num_cells_y_;
00401     Real delta_y_;
00402
00403     Real bottom_bndy_;
00404     Real top_bndy_;
00405     int num_cells_z_;
00406     Real delta_z_;
00407 };
00408 }
00409 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_3D_H_

```

18.87 README.md File Reference

18.88 README.md

```
00001 # The Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu**
00004
00005 ## 1. Description
00006
00007 We define numerical methods that are based on discretizations preserving the
00008 properties of their continuous counterparts to be **mimetic**.
00009
00010 The **Mimetic Methods Toolkit (MTK)** is a C++11 library for mimetic numerical
00011 methods. It is a set of classes for **mimetic interpolation**, **mimetic
00012 quadratures**, and **mimetic finite difference** methods for the **numerical
00013 solution of ordinary and partial differential equations**.
00014
00015 ## 2. Dependencies
00016
00017 This README file assumes all of these dependencies are installed in the
00018 following folder:
00019
00020 ``
00021 $(HOME)/Libraries/
00022 ``
00023
00024 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00025 routines for the internal computation on some of the layers. However, ATLAS
00026 requires both BLAS and LAPACK in order to create their optimized distributions.
00027 Therefore, the following dependencies tree arises:
00028
00029 ### For Linux:
00030
00031 1. LAPACK - Available from: http://www.netlib.org/lapack/
00032     1. BLAS - Available from: http://www.netlib.orgblas/
00033
00034 2. GLPK - Available from: https://www.gnu.org/software/glpk/
00035
00036 3. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00037     1. LAPACK - Available from: http://www.netlib.org/lapack/
00038     1. BLAS - Available from: http://www.netlib.orgblas/
00039
00040 4. (Optional) Valgrind - Available from: http://valgrind.org/
00041
00042 5. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00043
00044 ### For OSX:
00045
00046 1. GLPK - Available from: https://www.gnu.org/software/glpk/
00047
00048 ## 3. Installation
00049
00050 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00051
00052 The following steps are required to build and test the MTK. Please use the
00053 accompanying 'Makefile.inc' file, which should provide a solid template to
00054 start with. The following command provides help on the options for make:
00055 ``
00056 ``
00057 $ make help
00058 -----
00059 Makefile for the MTK.
00060
00061 Options are:
00062 - all: builds the library, the tests, and examples.
00063 - mtklib: builds the library.
00064 - test: builds the test files.
00065 - example: builds the examples.
00066
00067 - testall: runs all the tests.
00068
00069 - gendoc: generates the documentation for the library.
00070
00071 - clean: cleans all the generated files.
00072 - cleanlib: cleans the generated archive and object files.
00073 - cleantest: cleans the generated tests executables.
```

```

00074 - cleanexample: cleans the generated examples executables.
00075 -----
00076 ``
00077
00078 ### PART 2. BUILD THE LIBRARY.
00079
00080 ``
00081 $ make
00082 ``
00083
00084 If successful you'll read (before building the tests and examples):
00085 ``
00086 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00087 ``
00088
00089 ## 4. Contact, Support, and Credits
00090
00091 The GitHub repository is: https://github.com/ejspeiro/MTK
00092
00093 The MTK is developed by researchers and adjuncts to the
00094 [Computational Science Research Center (CSRC)](http://www.csrc.sdsu.edu/)
00095 at [San Diego State University (SDSU)](http://www.sdsu.edu/).
00096
00097 Currently the developers are:
00098
00099 - **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - @ejspeiro
00100 - Jose E. Castillo, PhD - jcastillo at mail dot sdsu dot edu
00101 - Guillermo F. Miranda, PhD - unigrav at hotmail dot com
00102
00103 ### 4.1. Acknowledgements and Contributions
00104
00105 The authors would like to acknowledge valuable advising, feedback,
00106 and actual contributions from research personnel at the Computational Science
00107 Research Center (CSRC) at San Diego State University (SDSU). Their input was
00108 important to the fruition of this work. Specifically, our thanks go to
00109 (alphabetical order):
00110
00111 - Mohammad Abouali, PhD
00112 - Dany De Cecchis, PhD
00113 - Otilio Rojas, PhD
00114 - Julia Rossi, PhD
00115 - Christopher P. Paolini, PhD - paolini at engineering dot sdsu dot edu
00116 - Johnny Corbino.
00117 - Raul Vargas-Navarro.
00118
00119 ## 5. Referencing This Work
00120
00121 Please reference this work as follows:
00122 ``
00123 @article{Sanchez2014308,
00124   title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
00125 Finite Differences",
00126   journal = "Journal of Computational and Applied Mathematics",
00127   volume = "270",
00128   number = "",
00129   pages = "308 - 322",
00130   year = "2014",
00131   note = "Fourth International Conference on Finite Element Methods in
00132 Engineering and Sciences (FEMTEC 2013)",
00133   issn = "0377-0427",
00134   doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
00135   url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
00136   author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
00137   keywords = "Object-oriented development",
00138   keywords = "Partial differential equations",
00139   keywords = "Application programming interfaces",
00140   keywords = "Mimetic Finite Differences"
00141 }
00142
00143 @Inbook{Sanchez2015,
00144   author = "Sanchez, Eduardo and Paolini, Christopher and Blomgren, Peter
00145 and Castillo, Jose",
00146   editor = "Kirby, M. Robert and Berzins, Martin and Hesthaven, S. Jan",
00147   chapter = "Algorithms for Higher-Order Mimetic Operators",
00148   titles = "Spectral and High Order Methods for Partial Differential Equations
00149 ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014,
00150 Salt Lake City, Utah, USA",
00151   year = "2015",
00152   publisher = "Springer International Publishing",
00153   address = "Cham",
00154   pages = "425--434",

```

```

00155     isbn="978-3-319-19800-2",
00156     doi="10.1007/978-3-319-19800-2_39",
00157     url="http://dx.doi.org/10.1007/978-3-319-19800-2_39"
00158 }
00159 ``
00160
00161 Finally, please free to contact me with suggestions or corrections:
00162
00163 **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - @ejspeiro
00164
00165 Thanks and happy coding!

```

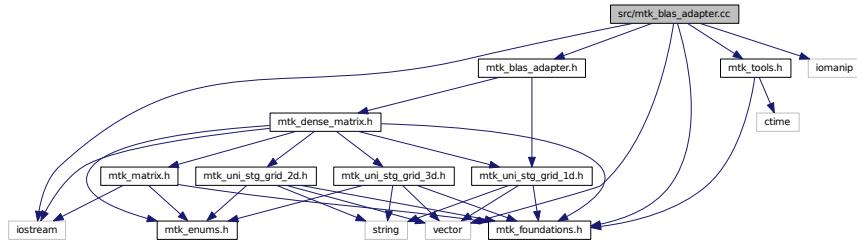
18.89 src/mtk blas_adapter.cc File Reference

Definition of an adapter class for the BLAS API.

```

#include <iostream>
#include <iomanip>
#include <vector>
#include "mtk_foundations.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::sgemm_](#) (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, int *ldb, double *beta, double *c, int *ldc)

18.89.1 Detailed Description

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

BLAS.

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

The BLAS can be installed from links given in the See Also section of this page.

See also

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

Todo Write documentation using LaTeX.

Author

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

Definition in file [mtk blas adapter.cc](#).

18.90 mtk blas adapter.cc

```

00001
00027 /*
00028 Copyright (C) 2016, Computational Science Research Center, San Diego State
00029 University. All rights reserved.
00030
00031 Redistribution and use in source and binary forms, with or without modification,
00032 are permitted provided that the following conditions are met:
00033
00034 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00035 and a copy of the modified files should be reported once modifications are
00036 completed, unless these modifications are made through the project's GitHub
00037 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00038 should be developed and included in any deliverable.
00039
00040 2. Redistributions of source code must be done through direct
00041 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00042
00043 3. Redistributions in binary form must reproduce the above copyright notice,
00044 this list of conditions and the following disclaimer in the documentation and/or
00045 other materials provided with the distribution.
00046
00047 4. Usage of the binary form on proprietary applications shall require explicit
00048 prior written permission from the the copyright holders, and due credit should
00049 be given to the copyright holders.
00050
00051 5. Neither the name of the copyright holder nor the names of its contributors
00052 may be used to endorse or promote products derived from this software without
00053 specific prior written permission.
00054
00055 The copyright holders provide no reassurances that the source code provided does
00056 not infringe any patent, copyright, or any other intellectual property rights of
00057 third parties. The copyright holders disclaim any liability to any recipient for
00058 claims brought against recipient by any third party for infringement of that
00059 parties intellectual property rights.
00060
00061 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00062 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00063 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00064 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00065 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00066 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00067 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00068 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00069 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00070 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```

```
00071 */
00072
00073 #include <iostream>
00074 #include <iomanip>
00075
00076 #include <vector>
00077
00078 #include "mtk_foundations.h"
00079 #include "mtk_tools.h"
00080 #include "mtk_blas_adapter.h"
00081
00082 namespace mtk {
00083
00084 extern "C" {
00085
00086 #ifdef MTK_PRECISION_DOUBLE
00087
00088 double dnrm2_(int *n, double *x, int *incx);
00089 #else
00090
00091 float snrm2_(int *n, float *x, int *incx);
00092 #endif
00093
00094 #ifdef MTK_PRECISION_DOUBLE
00095
00096 void daxpy_(int *n, double *da, double *dx, int *incx, double *dy, int *incy);
00097 #else
00098
00099 void saxpy_(int *n, float *sa, float *sx, int *incx, float *sy, int *incy);
00100 #endif
00101
00102 #ifdef MTK_PRECISION_DOUBLE
00103
00104 void dgemv_(char *trans,
00105             int *m,
00106             int *n,
00107             double *alpha,
00108             double *a,
00109             int *lda,
00110             double *x,
00111             int *incx,
00112             double *beta,
00113             double *y,
00114             int *incy);
00115 #else
00116
00117 void sgemv_(char *trans,
00118             int *m,
00119             int *n,
00120             float *alpha,
00121             float *a,
00122             int *lda,
00123             float *x,
00124             int *incx,
00125             float *beta,
00126             float *y,
00127             int *incy);
00128 #endif
00129
00130 #ifdef MTK_PRECISION_DOUBLE
00131
00132 void dgemm_(char *transa,
00133               char* transb,
00134               int *m,
00135               int *n,
00136               int *k,
00137               double *alpha,
00138               double *a,
00139               int *lda,
00140               double *b,
00141               int *ldb,
00142               double *beta,
00143               double *c,
00144               int *ldc);
00145 }
00146 #else
00147
00148 void sgemm_(char *transa,
00149               char* transb,
00150               int *m,
00151               int *n,
```

```

00314         int *k,
00315         double *alpha,
00316         double *a,
00317         int *ida,
00318         double *b,
00319         int *idb,
00320         double *beta,
00321         double *c,
00322         int *ldc);
00323 }
00324 #endif
00325 }
00326
00327 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00328
00329 #ifdef MTK_PERFORM_PREVENTIONS
00330     mtk::Tools::Prevent(in_length <= 0, __FILE__, __LINE__, __func__);
00331 #endif
00332
00333 int incx{1}; // Increment for the elements of xx. ix >= 0.
00334
00335 #ifdef MTK_PRECISION_DOUBLE
00336     return dnrm2_(&in_length, in, &incx);
00337 #else
00338     return snrm2_(&in_length, in, &incx);
00339 #endif
00340 }
00341
00342 void mtk::BLASAdapter::RealAXPY(mtk::Real alpha,
00343                                     mtk::Real *xx,
00344                                     mtk::Real *yy,
00345                                     int &in_length) {
00346
00347 #ifdef MTK_PERFORM_PREVENTIONS
00348     mtk::Tools::Prevent(xx == nullptr, __FILE__, __LINE__, __func__);
00349     mtk::Tools::Prevent(yy == nullptr, __FILE__, __LINE__, __func__);
00350 #endif
00351
00352 int incx{1}; // Increment for the elements of xx. ix >= 0.
00353
00354 #ifdef MTK_PRECISION_DOUBLE
00355     daxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00356 #else
00357     saxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00358 #endif
00359 }
00360
00361 mtk::Real mtk::BLASAdapter::RelNorm2Error(
00362     mtk::Real *computed,
00363                                     mtk::Real *known,
00364                                     int length) {
00365
00366 #ifdef MTK_PERFORM_PREVENTIONS
00367     mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00368     mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00369 #endif
00370     mtk::Real norm_2_computed{mtk::BLASAdapter::RealNRM2(known, length)};
00371
00372     mtk::Real alpha{-mtk::kOne};
00373
00374     mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00375
00376     mtk::Real norm_2_difference{mtk::BLASAdapter::RealNRM2(computed,
00377                                         length)};
00378
00379     return norm_2_difference/norm_2_computed;
00380 }
00381
00382 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00383                                     mtk::DenseMatrix &aa,
00384                                     mtk::Real *xx,
00385                                     mtk::Real &beta,
00386                                     mtk::Real *yy) {
00387
00388 // Make sure input matrices are row-major ordered.
00389 if (aa.matrix_properties().ordering() ==
00390     mtk::MatrixOrdering::COL_MAJOR) {
00391     aa.OrderRowMajor();
00391 }

```

```

00392
00393     char transa('T'); // State that now, the input WILL be in row-major ordering.
00394
00395     int mm{aa.num_rows()}; // Rows of aa.
00396     int nn{aa.num_cols()}; // Columns of aa.
00397     int lda{(aa.matrix_properties()).leading_dimension()}; // Leading dimension.
00398     int incx{1}; // Increment x vals.
00399     int incy{1}; // Increment y vals.
00400
00401     std::swap(mm,nn);
00402 #ifdef MTK_PRECISION_DOUBLE
00403     dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00404             xx, &incx, &beta, yy, &incy);
00405 #else
00406     sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00407             xx, &incx, &beta, yy, &incy);
00408 #endif
00409     std::swap(mm,nn);
00410 }
00411
00412 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00413                                         mtk::Real *aa,
00414                                         mtk::MatrixOrdering &ordering,
00415                                         int num_rows,
00416                                         int num_cols,
00417                                         int lda,
00418                                         mtk::Real *xx,
00419                                         mtk::Real &beta,
00420                                         mtk::Real *yy) {
00421
00422     // Make sure input matrices are row-major ordered.
00423
00424 #ifdef MTK_PERFORM_PREVENTIONS
00425     mtk::Tools::Prevent(ordering !=
00426                         mtk::MatrixOrdering::ROW_MAJOR,
00427                         __FILE__, __LINE__, __func__);
00428
00429
00430     char transa('T'); // State that now, the input WILL be in row-major ordering.
00431
00432     int mm{num_rows}; // Rows of aa.
00433     int nn{num_cols}; // Columns of aa.
00434     int incx{1}; // Increment of values in x.
00435     int incy{1}; // Increment of values in y.
00436
00437     std::swap(mm,nn);
00438 #ifdef MTK_PRECISION_DOUBLE
00439     dgemv_(&transa, &mm, &nn, &alpha, aa, &lda, xx, &incx, &beta, yy, &incy);
00440 #else
00441     sgemv_(&transa, &mm, &nn, &alpha, aa, &lda, xx, &incx, &beta, yy, &incy);
00442 #endif
00443     std::swap(mm,nn);
00444 }
00445
00446 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
00447     mtk::DenseMatrix &aa,
00448                                         mtk::DenseMatrix &bb) {
00449
00450 #ifdef MTK_PERFORM_PREVENTIONS
00451     mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00452                         __FILE__, __LINE__, __func__);
00453
00454
00455     if (aa.matrix_properties().ordering() ==
00456         mtk::MatrixOrdering::COL_MAJOR) {
00457         aa.OrderRowMajor();
00458     }
00459     if (bb.matrix_properties().ordering() ==
00460         mtk::MatrixOrdering::COL_MAJOR) {
00461         bb.OrderRowMajor();
00462     }
00463
00464     char ta('T'); // State that input matrix aa is in row-wise ordering.
00465     char tb('T'); // State that input matrix bb is in row-wise ordering.
00466
00467     int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00468     int nn{bb.num_cols()}; // Cols of bb and cols of cc.
00469     int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00470
00471     int cc_num_rows{mm}; // Rows of cc.

```

```

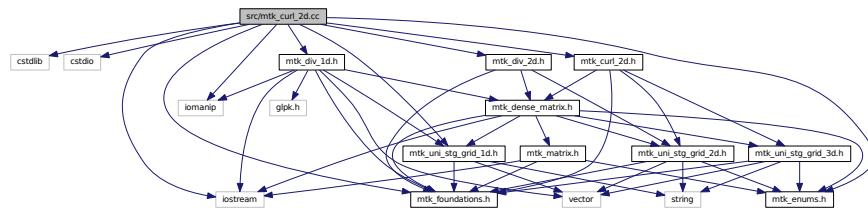
00471 int cc_num_cols(nn); // Columns of cc.
00472
00473 int lda{std::max(1,kk)}; // Leading dimension of the aa matrix.
00474 int ldb{std::max(1,nn)}; // Leading dimension of the bb matrix.
00475 int ldc{std::max(1,mm)}; // Leading dimension of the cc matrix.
00476
00477 mtk::Real alpha{mtk::kOne}; // First scalar coefficient.
00478 mtk::Real beta{mtk::kZero}; // Second scalar coefficient.
00479
00480 mtk::DenseMatrix cc_col_maj_ord(cc_num_rows,cc_num_cols); // Output matrix.
00481
00482 cc_col_maj_ord.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00483
00484 #ifdef MTK_PRECISION_DOUBLE
00485 dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00486         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00487 #else
00488 sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00489         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00490 #endif
00491
00492 #if MTK_VERBOSE_LEVEL > 12
00493 std::cout << "cc_col_maj_ord =" << std::endl;
00494 std::cout << cc_col_maj_ord << std::endl;
00495 #endif
00496
00497 cc_col_maj_ord.OrderRowMajor();
00498
00499 return cc_col_maj_ord;
00500 }
00501
00502 mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM(
00503     mtk::Real alpha,
00504                                         mtk::DenseMatrix &aa) {
00505
00506 #ifdef MTK_PERFORM_PREVENTIONS
00507 mtk::Tools::Prevent(aa.num_rows() == 0, __FILE__, __LINE__, __func__);
00508 mtk::Tools::Prevent(aa.num_cols() == 0, __FILE__, __LINE__, __func__);
00509 #endif
00510
00511 if (aa.matrix_properties().ordering() ==
00512     mtk::MatrixOrdering::COL_MAJOR) {
00513     aa.OrderRowMajor();
00514 }
00515
00516 char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00517 char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00518
00519 int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00520 int nn{aa.num_cols()}; // Cols of bb and cols of cc.
00521 int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00522
00523 int lda{std::max(1,kk)}; // Leading dimension of the aa matrix.
00524 int ldb{std::max(1,nn)}; // Leading dimension of the bb matrix.
00525 int ldc{std::max(1,mm)}; // Leading dimension of the cc matrix.
00526
00527 mtk::Real beta{alpha}; // Second scalar coefficient.
00528
00529 alpha = mtk::kZero;
00530
00531 mtk::DenseMatrix alpha_aa(aa); // Output matrix.
00532
00533 #ifdef MTK_PRECISION_DOUBLE
00534 dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00535         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00536 #else
00537 sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00538         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00539 #endif
00540
00541 #if MTK_VERBOSE_LEVEL > 12
00542 std::cout << "alpha_aa =" << std::endl;
00543 std::cout << alpha_aa << std::endl;
00544 #endif
00545
00546 return alpha_aa;
00547
00548 }
00549 }
```

18.91 src/mtk_curl_2d.cc File Reference

Implements the class `Curl2D`.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"
```

Include dependency graph for mtk_curl_2d.cc:



18.91.1 Detailed Description

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

Author

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

Definition in file [mtk_curl_2d.cc](#).

18.92 mtk_curl_2d.cc

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
```

```

00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtkEnums.h"
00065 #include "mtkUniStgGrid1d.h"
00066 #include "mtkDiv1d.h"
00067 #include "mtkDiv2d.h"
00068 #include "mtkCurl2d.h"
00069
00070 mtk::UniStgGrid3D mtk::Curl2D::operator*(const
    mtk::UniStgGrid2D &grid) const {
00071
00072
00073     mtk::UniStgGrid3D output;
00074
00075     return output;
00076 }
00077
00078
00079 mtk::Curl2D::Curl2D():
00080     order_accuracy_(),
00081     mimetic_threshold_() {}
00082
00083 mtk::Curl2D::Curl2D(const Curl2D &curl):
00084     order_accuracy_(curl.order_accuracy_),
00085     mimetic_threshold_(curl.mimetic_threshold_) {}
00086
00087 mtk::Curl2D::~Curl2D() {}
00088
00089 bool mtk::Curl2D::ConstructCurl2D(const
    mtk::UniStgGrid2D &grid,
00090                                         int order_accuracy,
00091                                         mtk::Real mimetic_threshold) {
00092
00093     int num_cells_x = grid.num_cells_x();
00094     int num_cells_y = grid.num_cells_y();
00095
00096     int mx = num_cells_x + 2; // Dx vertical dimension.
00097     int nx = num_cells_x + 1; // Dx horizontal dimension.
00098     int my = num_cells_y + 2; // Dy vertical dimension.
00099     int ny = num_cells_y + 1; // Dy horizontal dimension.
00100
00101     mtk::Div1D div;
00102
00103     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00104
00105     #ifdef MTK_PERFORM_PREVENTIONS
00106     if (!info) {
00107         std::cerr << "Mimetic div could not be built." << std::endl;
00108         return info;
00109     }
00110     #endif

```

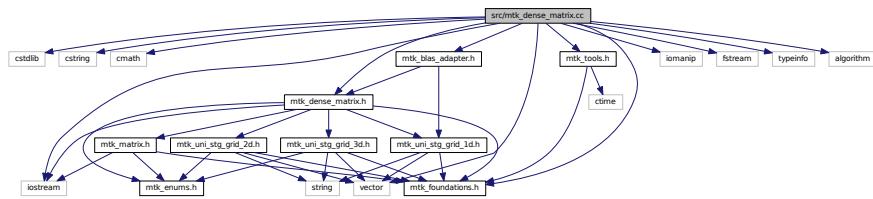
```

00111    auto west = grid.west_bndy();
00112    auto east = grid.east_bndy();
00113    auto south = grid.south_bndy();
00114    auto north = grid.east_bndy();
00115
00116
00117    mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00118    mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00119
00120    mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00121    mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00122
00123    bool padded{true};
00124    bool transpose{false};
00125
00126    mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00127    mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00128
00129    mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00130    mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00131
00132 #if MTK_VERBOSE_LEVEL > 2
00133 std::cout << "Dx: " << mx << " by " << nx << std::endl;
00134 std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;
00135 std::cout << "Dy: " << my << " by " << ny << std::endl;
00136 std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00137 std::cout << "Curl 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00138     nx*ny << std::endl;
00139 #endif
00140
00141    mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00142
00143    for (auto ii = 0; ii < mx*my; ii++) {
00144        for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00145            d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00146        }
00147        for (auto kk = 0; kk < ny*num_cells_x; kk++) {
00148            d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00149        }
00150    }
00151
00152    curl_ = d2d;
00153
00154    curl_.set_encoded_operator(mtk::EncodedOperator::CURLE);
00155
00156    return info;
00157 }
00158
00159 mtk::DenseMatrix mtk::Curl2D::ReturnAsDenseMatrix() const {
00160
00161     return curl_;
00162 }
```

18.93 src/mtk_dense_matrix.cc File Reference

```
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <typeinfo>
#include <vector>
#include <algorithm>
#include "mtk_foundations.h"
#include "mtk_tools.h"
#include "mtk_dense_matrix.h"
#include "mtk blas_adapter.h"
```

Include dependency graph for mtk_dense_matrix.cc:



Namespaces

- **mtk**

Mimetic Methods Toolkit namespace.

Functions

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

18.94 mtk_dense_matrix.cc

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
  
```

```

00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstring>
00059 #include <cmath>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063 #include <fstream>
00064
00065 #include <typeinfo>
00066
00067 #include <vector>
00068
00069 #include <algorithm>
00070
00071 #include "mtk_foundations.h"
00072 #include "mtk_tools.h"
00073 #include "mtk_dense_matrix.h"
00074 #include "mtk blas_adapter.h"
00075
00076 namespace mtk {
00077
00078 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00079
00080     int mm{in.matrix_properties_.num_rows()}; // Auxiliary.
00081     int nn{in.matrix_properties_.num_cols()}; // Auxiliary.
00082     int output_precision{4};
00083     int output_width{10};
00084
00085     if (in.matrix_properties_.ordering() ==
00086         mtk::MatrixOrdering::COL_MAJOR) {
00086         std::swap(mm, nn);
00087     }
00088     for (int ii = 0; ii < mm; ii++) {
00089         int offset{ii*nn};
00090         for (int jj = 0; jj < nn; jj++) {
00091             mtk::Real value = in.data_[offset + jj];
00092             stream << std::setprecision(output_precision) <<
00093                 std::setw(output_width) << value;
00094         }
00095         stream << std::endl;
00096     }
00097     if (in.matrix_properties_.ordering() ==
00098         mtk::MatrixOrdering::COL_MAJOR) {
00098         std::swap(mm, nn);
00099     }
00100     return stream;
00101 }
00102 }
00103
00104 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
00105     mtk::DenseMatrix &in) {
00106
00107     if(this == &in) {
00108         return *this;
00109     }
00110
00111     matrix_properties_.set_encoded_operator(
00112         in.matrix_properties_.encoded_operator());
00113
00114     matrix_properties_.set_storage(in.
00115         matrix_properties_.storage());
00116
00117     auto aux = in.matrix_properties_.num_rows();
00118     matrix_properties_.set_num_rows(aux);
00119
00120     aux = in.matrix_properties().num_cols();
00121     matrix_properties_.set_num_cols(aux);
00122
00123     aux = in.matrix_properties().num_zero();
00124     matrix_properties_.set_num_zero(aux);
00125
00126     aux = in.matrix_properties().num_null();
00127     matrix_properties_.set_num_null(aux);

```

```

00128
00129     auto num_rows = matrix_properties_.num_rows();
00130     auto num_cols = matrix_properties_.num_cols();
00131
00132     delete [] data_;
00133
00134     try {
00135         data_ = new mtk::Real[num_rows*num_cols];
00136     } catch (std::bad_alloc &memory_allocation_exception) {
00137         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00138         std::endl;
00139         std::cerr << memory_allocation_exception.what() << std::endl;
00140     }
00141     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
00142         num_cols);
00143     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00144
00145     return *this;
00146 }
00147
00148 bool mtk::DenseMatrix::operator==(const
00149 DenseMatrix &in) {
00150     bool ans{true};
00151
00152     ans = ans &&
00153         (matrix_properties_.encoded_operator() ==
00154             matrix_properties_.encoded_operator());
00155
00156     auto mm = in.num_rows();
00157     auto nn = in.num_cols();
00158
00159     if (mm != matrix_properties_.num_rows() ||
00160         nn != matrix_properties_.num_cols()) {
00161         return false;
00162     }
00163
00164     for (int ii = 0; ii < mm && ans; ++ii) {
00165         for (int jj = 0; jj < nn && ans; ++jj) {
00166             ans = ans &&
00167                 abs(data_[ii*nn + jj] - in.data()[ii*nn + jj]) <
00168                     mtk::kDefaultTolerance;
00169         }
00170     }
00171     return ans;
00172 }
00173 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00174
00175     matrix_properties_.set_encoded_operator(
00176         mtk::EncodedOperator::NOOP);
00177
00178     matrix_properties_.set_storage(
00179         mtk::MatrixStorage::DENSE);
00180     matrix_properties_.set_ordering(
00181         mtk::MatrixOrdering::ROW_MAJOR);
00182
00183     matrix_properties_.set_encoded_operator(
00184         in.matrix_properties_.encoded_operator());
00185
00186     matrix_properties_.set_storage(in.matrix_properties_.storage());
00187
00188     matrix_properties_.set_ordering(in.matrix_properties_.
00189         ordering());
00190
00191     auto aux = in.matrix_properties_.num_rows();
00192     matrix_properties_.set_num_rows(aux);
00193
00194     aux = in.matrix_properties_.num_cols();
00195     matrix_properties_.set_num_cols(aux);
00196
00197     aux = in.matrix_properties_.num_zero();
00198     matrix_properties_.set_num_zero(aux);
00199
00200     aux = in.matrix_properties_.num_null();
00201     matrix_properties_.set_num_null(aux);

```

```

00201     auto num_rows = in.matrix_properties_.num_rows();
00202     auto num_cols = in.matrix_properties_.num_cols();
00203
00204     try {
00205         data_ = new mtk::Real[num_rows*num_cols];
00206     } catch (std::bad_alloc &memory_allocation_exception) {
00207         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00208             std::endl;
00209         std::cerr << memory_allocation_exception.what() << std::endl;
00210     }
00211     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00212     std::copy(in.data_, in.data_ + num_rows*num_cols,data_);
00213
00214 }
00215
00216 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00217
00218 #ifdef MTK_PERFORM_PREVENTIONS
00219     mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00220     mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00221 #endif
00222
00223     matrix_properties_.set_encoded_operator(mtk::EncodedOperator::NOOP);
00224
00225     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00226     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00227     matrix_properties_.set_num_rows(num_rows);
00228     matrix_properties_.set_num_cols(num_cols);
00229
00230     try {
00231         data_ = new mtk::Real[num_rows*num_cols];
00232     } catch (std::bad_alloc &memory_allocation_exception) {
00233         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00234             std::endl;
00235         std::cerr << memory_allocation_exception.what() << std::endl;
00236     }
00237     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00238 }
00239
00240 mtk::DenseMatrix::DenseMatrix(const int &rank,
00241                               const bool &padded,
00242                               const bool &transpose) {
00243
00244 #ifdef MTK_PERFORM_PREVENTIONS
00245     mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00246 #endif
00247
00248     int aux{}; // Used to control the padding.
00249
00250     if (padded) {
00251         aux = 1;
00252     }
00253
00254     matrix_properties_.set_encoded_operator(mtk::EncodedOperator::NOOP);
00255
00256     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00257     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00258     matrix_properties_.set_num_rows(aux + rank + aux);
00259     matrix_properties_.set_num_cols(rank);
00260
00261     try {
00262         data_ = new mtk::Real[matrix_properties_.num_values()];
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_,
00269            mtk::kZero,
00270            sizeof(data_[0])*(matrix_properties_.num_values()));
00271
00272     for (auto ii = 0 ; ii < matrix_properties_.num_rows(); ++ii) {
00273         for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00274             data_[ii*matrix_properties_.num_cols() + jj] =
00275                 (ii == jj + aux)? mtk::kOne: mtk::kZero;
00276         }
00277     }
00278
00279     if (transpose) {
00280         Transpose();
00281     }

```

```

00282 }
00283
00284 mtk::DenseMatrix::DenseMatrix(const mtk::Real *const gen,
00285                                     const int &gen_length,
00286                                     const int &pro_length,
00287                                     const bool &transpose) {
00288
00289 #ifdef MTK_PERFORM_PREVENTIONS
00290     mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00291     mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00292     mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00293 #endif
00294
00295     matrix_properties_.set_encoded_operator(mtk::EncodedOperator::NOOP);
00296
00297     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00298     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00299     if (!transpose) {
00300         matrix_properties_.set_num_rows(gen_length);
00301         matrix_properties_.set_num_cols(pro_length);
00302     } else {
00303         matrix_properties_.set_num_rows(pro_length);
00304         matrix_properties_.set_num_cols(gen_length);
00305     }
00306
00307     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00308     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00309
00310     try {
00311         data_ = new mtk::Real[mm*nn];
00312     } catch (std::bad_alloc &memory_allocation_exception) {
00313         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00314             std::endl;
00315         std::cerr << memory_allocation_exception.what() << std::endl;
00316     }
00317     memset(data_, mtk::kZero, sizeof(data_[0])*mm*nn);
00318
00319     if (!transpose) {
00320         for (auto ii = 0; ii < mm; ii++) {
00321             for (auto jj = 0; jj < nn; jj++) {
00322                 data_[ii*nn + jj] = pow(gen[ii], (double) jj);
00323             }
00324         }
00325     } else {
00326         for (auto ii = 0; ii < mm; ii++) {
00327             for (auto jj = 0; jj < nn; jj++) {
00328                 data_[ii*nn + jj] = pow(gen[jj], (double) ii);
00329             }
00330         }
00331     }
00332 }
00333
00334 mtk::DenseMatrix::~DenseMatrix() {
00335
00336     delete [] data_;
00337     data_ = nullptr;
00338 }
00339
00340 mtk::Matrix mtk::DenseMatrix::matrix_properties() const
00341     noexcept {
00342
00343     return matrix_properties_;
00344 }
00345
00346 void mtk::DenseMatrix::SetOrdering(
00347     mtk::MatrixOrdering oo) noexcept {
00348
00349 #ifdef MTK_PERFORM_PREVENTIONS
00350     mtk::Tools::Prevent(!!(oo == mtk::MatrixOrdering::ROW_MAJOR
00351 || oo ==
00352 mtk::MatrixOrdering::COL_MAJOR),
00353             __FILE__, __LINE__, __func__);
00354 #endif
00355
00356     matrix_properties_.set_ordering(oo);
00357 }
00358
00359 int mtk::DenseMatrix::num_rows() const noexcept {
00360
00361     return matrix_properties_.num_rows();
00362 }
```

```

00360
00361 int mtk::DenseMatrix::num_cols() const noexcept {
00362
00363     return matrix_properties_.num_cols();
00364 }
00365
00366 mtk::Real* mtk::DenseMatrix::data() const noexcept {
00367
00368     return data_;
00369 }
00370
00371 mtk::EncodedOperator mtk::DenseMatrix::encoded_operator
00372     () const {
00373
00374     return matrix_properties_.encoded_operator();
00375 }
00376 void mtk::DenseMatrix::set_encoded_operator(const
00377     EncodedOperator &op) {
00378
00379 #ifdef MTK_PERFORM_PREVENTIONS
00380     bool aux = (op != mtk::EncodedOperator::NOOP) &&
00381         (op != mtk::EncodedOperator::GRADIENT) &&
00382         (op != mtk::EncodedOperator::DIVERGENCE) &&
00383         (op != mtk::EncodedOperator::INTERPOLATION) &&
00384         (op != mtk::EncodedOperator::CURL) &&
00385         (op != mtk::EncodedOperator::LAPLACIAN);
00386
00387     mtk::Tools::Prevent(aux, __FILE__, __LINE__, __func__);
00388 #endif
00389
00390     matrix_properties_.set_encoded_operator(op);
00391 }
00392 mtk::Real mtk::DenseMatrix::GetValue(
00393     const int &mm,
00394     const int &nn) const noexcept {
00395
00396 #ifdef MTK_PERFORM_PREVENTIONS
00397     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00398     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00399 #endif
00400
00401     return data_[mm*matrix_properties_.num_cols() + nn];
00402 }
00403
00404 void mtk::DenseMatrix::SetValue(
00405     const int &mm,
00406     const int &nn,
00407     const mtk::Real &val) noexcept {
00408
00409 #ifdef MTK_PERFORM_PREVENTIONS
00410     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00411     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00412 #endif
00413
00414     data_[mm*matrix_properties_.num_cols() + nn] = val;
00415 }
00416
00417 void mtk::DenseMatrix::Transpose() {
00418
00419     mtk::Real *data_transposed{}; // Buffer.
00420
00421     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00422     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00423
00424     try {
00425         data_transposed = new mtk::Real[mm*nn];
00426     } catch (std::bad_alloc &memory_allocation_exception) {
00427         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00428             std::endl;
00429         std::cerr << memory_allocation_exception.what() << std::endl;
00430     }
00431     memset(data_transposed,
00432             mtk::kZero,
00433             sizeof(data_transposed[0])*mm*nn);
00434
00435     // Assign the values to their transposed position.
00436     for (auto ii = 0; ii < mm; ++ii) {
00437         for (auto jj = 0; jj < nn; ++jj) {
00438
00439

```

```

00440     data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00441 }
00442 }
00443
00444 // Swap pointers.
00445 auto tmp = data_; // Temporal holder.
00446 data_ = data_transposed;
00447 delete [] tmp;
00448 tmp = nullptr;
00449
00450 matrix_properties_.set_num_rows(nn);
00451 matrix_properties_.set_num_cols(mm);
00452 }
00453
00454 void mtk::DenseMatrix::OrderRowMajor() {
00455
00456 if (matrix_properties_.ordering() == mtk::MatrixOrdering::COL_MAJOR) {
00457
00458
00459
00460     mtk::Real *data_transposed{}; // Buffer.
00461
00462     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00463     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00464
00465     try {
00466         data_transposed = new mtk::Real[mm*nn];
00467     } catch (std::bad_alloc &memory_allocation_exception) {
00468         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00469             std::endl;
00470         std::cerr << memory_allocation_exception.what() << std::endl;
00471     }
00472     memset(data_transposed,
00473             mtk::kZero,
00474             sizeof(data_transposed[0])*mm*nn);
00475
00476 // Assign the values to their transposed position.
00477 std::swap(mm, nn);
00478 for (auto ii = 0; ii < mm; ++ii) {
00479     for (auto jj = 0; jj < nn; ++jj) {
00480         data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00481     }
00482 }
00483 std::swap(mm, nn);
00484
00485 // Swap pointers.
00486 auto tmp = data_; // Temporal holder.
00487 data_ = data_transposed;
00488 delete [] tmp;
00489 tmp = nullptr;
00490
00491     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00492 }
00493 }
00494
00495 void mtk::DenseMatrix::OrderColMajor() {
00496
00497 if (matrix_properties_.ordering() == mtk::MatrixOrdering::ROW_MAJOR) {
00498
00499
00500
00501     mtk::Real *data_transposed{}; // Buffer.
00502
00503     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00504     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00505
00506     try {
00507         data_transposed = new mtk::Real[mm*nn];
00508     } catch (std::bad_alloc &memory_allocation_exception) {
00509         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00510             std::endl;
00511         std::cerr << memory_allocation_exception.what() << std::endl;
00512     }
00513     memset(data_transposed,
00514             mtk::kZero,
00515             sizeof(data_transposed[0])*mm*nn);
00516
00517 // Assign the values to their transposed position.
00518 for (auto ii = 0; ii < mm; ++ii) {
00519     for (auto jj = 0; jj < nn; ++jj) {
00520         data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00521     }
00522 }

```

```

00523
00524     // Swap pointers.
00525     auto tmp = data_; // Temporal holder.
00526     data_ = data_transposed;
00527     delete [] tmp;
00528     tmp = nullptr;
00529
00530     matrix_properties_.set_ordering(mtk::MatrixOrdering::COL_MAJOR);
00531 }
00532 }
00533
00534 mtk::DenseMatrix mtk::DenseMatrix::Kron(const
00535     mtk::DenseMatrix &aa,
00536                                     const mtk::DenseMatrix &bb) {
00537
00538
00539     int row_offset{}; // Offset for rows.
00540     int col_offset{}; // Offset for rows.
00541
00542     mtk::Real aa_factor{}; // Used in computation.
00543
00544     // Auxiliary variables:
00545     auto aux1 = aa.matrix_properties_.num_rows()*bb.
00546     matrix_properties_.num_rows();
00547     auto aux2 = aa.matrix_properties_.num_cols()*bb.
00548     matrix_properties_.num_cols();
00549
00550     mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00551
00552     int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00553
00554     auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00555     auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00556     auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00557     auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00558
00559     for (auto ii = 0; ii < mm; ++ii) {
00560         row_offset = ii*pp;
00561         for (auto jj = 0; jj < nn; ++jj) {
00562             col_offset = jj*qq;
00563             aa_factor = aa.data_[ii*nn + jj];
00564             for (auto ll = 0; ll < pp; ++ll) {
00565                 for (auto oo = 0; oo < qq; ++oo) {
00566                     auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00567                     output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00568                 }
00569             }
00570         }
00571         output.matrix_properties_.set_storage(
00572             mtk::MatrixStorage::DENSE);
00573         output.matrix_properties_.set_ordering(
00574             mtk::MatrixOrdering::ROW_MAJOR);
00575     }
00576
00577     bool mtk::DenseMatrix::WriteToFile(const std::string &filename) const {
00578
00579     std::ofstream output_dat_file; // Output file.
00580
00581     output_dat_file.open(filename);
00582
00583     if (!output_dat_file.is_open()) {
00584         return false;
00585     }
00586
00587     int mm{matrix_properties_.num_rows()};
00588     int nn{matrix_properties_.num_cols()};
00589
00590     for (int ii = 0; ii < mm; ++ii) {
00591         int offset{ii*nn};
00592         for (int jj = 0; jj < nn; ++jj) {
00593             output_dat_file << ii << ' ' << jj << ' ' << data_[offset + jj] <<
00594             std::endl;
00595         }
00596     }
00597
00598     output_dat_file.close();
00599

```

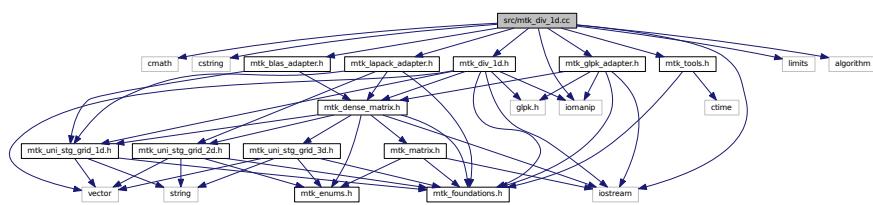
```
00600     return true;
00601 }
```

18.95 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)

18.95.1 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](#).

18.96 mtk_div_1d.cc

```
00001
00015 /*
00016 Copyright (C) 2016, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075 #include "mtk blas adapter.h"
00076 #include "mtk lapack adapter.h"
00077 #include "mtk glpk adapter.h"
00078 #include "mtk div_1d.h"
00079
00080 namespace mtk {
00081
00082 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00083
00084     int output_precision{4};
00085     int output_width{8};
00086 }
```

```

00088     stream << "Order of accuracy: " << in.divergence_[0] << std::endl;
00090
00092
00093     stream << "Interior stencil: " << std::endl;
00094     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00095         stream << std::setprecision(output_precision) << std::setw(output_width) <<
00096             in.divergence_[ii] << ' ';
00097     }
00098     stream << std::endl;
00099
00100    if (in.order_accuracy_ > 2) {
00101
00103
00104        stream << "Weights:" << std::endl;
00105        for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00106            order_accuracy_; ++ii) {
00107            stream << std::setprecision(output_precision) <<
00108                std::setw(output_width) << in.divergence_[ii] << ' ';
00109        }
00110        stream << std::endl;
00112
00113        auto offset = (2*in.order_accuracy_ + 1);
00114        int mm{};
00115        for (auto ii = 0; ii < in.dim_null_; ++ii) {
00116            stream << "Boundary row " << ii + 1 << ":" << std::endl;
00117            for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {
00118                auto value = in.divergence_[offset + mm];
00119                stream << std::setprecision(output_precision) <<
00120                    std::setw(output_width) << value << ' ';
00121                ++mm;
00122            }
00123            stream << std::endl;
00124            stream << "Sum of elements in boundary row " << ii + 1 << ": " <<
00125            in.sums_rows_mim_bndy_[ii];
00126            stream << std::endl;
00127        }
00128    }
00129
00130    return stream;
00131 }
00132 }
00133
00134 mtk::Div1D::Div1D():
00135     order_accuracy_(mtk::kDefaultOrderAccuracy),
00136     dim_null_(),
00137     num_bndy_coeffs_(),
00138     divergence_length_(),
00139     minrow_(),
00140     row_(),
00141     num_feasible_sols_(),
00142     coeffs_interior_(),
00143     prem_apps_(),
00144     weights_crs_(),
00145     weights_cbs_(),
00146     mim_bndy_(),
00147     divergence_(),
00148     mimetic_threshold_(mtk::kDefaultMimeticThreshold),
00149     mimetic_measure_(mtk::kZero),
00150     sums_rows_mim_bndy_(){}
00151
00152 mtk::Div1D::Div1D(const Div1D &div):
00153     order_accuracy_(div.order_accuracy_),
00154     dim_null_(div.dim_null_),
00155     num_bndy_coeffs_(div.num_bndy_coeffs_),
00156     divergence_length_(div.divergence_length_),
00157     minrow_(div.minrow_),
00158     row_(div.row_),
00159     num_feasible_sols_(div.num_feasible_sols_),
00160     coeffs_interior_(div.coeffs_interior_),
00161     prem_apps_(div.prem_apps_),
00162     weights_crs_(div.weights_crs_),
00163     weights_cbs_(div.weights_cbs_),
00164     mim_bndy_(div.mim_bndy_),
00165     divergence_(div.divergence_),
00166     mimetic_threshold_(div.mimetic_threshold_),
00167     mimetic_measure_(div.mimetic_measure_),
00168     sums_rows_mim_bndy_(div.sums_rows_mim_bndy_) {}
00169
00170 mtk::Div1D::~Div1D() {

```

```

00171     delete[] coeffs_interior_;
00172     coeffs_interior_ = nullptr;
00173
00174
00175     delete[] prem_apps_;
00176     prem_apps_ = nullptr;
00177
00178     delete[] weights_crs_;
00179     weights_crs_ = nullptr;
00180
00181     delete[] weights_cbs_;
00182     weights_cbs_ = nullptr;
00183
00184     delete[] mim_bndy_;
00185     mim_bndy_ = nullptr;
00186
00187     delete[] divergence_;
00188     divergence_ = nullptr;
00189 }
00190
00191 bool mtk::Div1D::ConstructDiv1D(int order_accuracy,
00192                                     mtk::Real mimetic_threshold) {
00193
00194 #ifdef MTK_PERFORM_PREVENTIONS
00195     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00196     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00197     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00198                         __FILE__, __LINE__, __func__);
00199
00200     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00201         std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00202     }
00203
00204     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00205     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00206 #endif
00207
00208     order_accuracy_ = order_accuracy;
00209     mimetic_threshold_ = mimetic_threshold;
00210
00211
00212
00213     bool abort_construction = ComputeStencilInteriorGrid();
00214
00215 #ifdef MTK_PERFORM_PREVENTIONS
00216     if (!abort_construction) {
00217         std::cerr << "Could NOT complete stage 1." << std::endl;
00218         std::cerr << "Exiting..." << std::endl;
00219         return false;
00220     }
00221 #endif
00222
00223 // At this point, we already have the values for the interior stencil stored
00224 // in the coeffs_interior_ array.
00225
00226 // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00227 // approximation at the boundary, thus it has no weights. For this case, the
00228 // dimension of the null-space of the Vandermonde matrices used to compute the
00229 // approximating coefficients at the boundary is 0. Ergo, we compute this
00230 // number first and then decide if we must compute anything at the boundary.
00231
00232     dim_null_ = order_accuracy_/2 - 1;
00233
00234     if (dim_null_ > 0) {
00235
00236 #ifdef MTK_PRECISION_DOUBLE
00237     num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00238 #else
00239     num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00240 #endif
00241
00242
00243     // For this we will follow recommendations given in:
00244     //
00245     // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00246     //
00247     // We will compute the QR Factorization of the transpose, as in the
00248     // following (MATLAB) pseudo-code:
00249     //
00250     // [Q,R] = qr(V'); % Full QR as defined in
00251     // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00252     //
00253

```

```

00254 // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00255 //
00256 // However, given the nature of the Vandermonde matrices we've just
00257 // computed, they all posses the same null-space. Therefore, we impose the
00258 // convention of computing the null-space of the first Vandermonde matrix
00259 // (west boundary).
00260
00261 abort_construction = ComputeRationalBasisNullSpace();
00262
00263 #ifdef MTK_PERFORM_PREVENTIONS
00264 if (!abort_construction) {
00265     std::cerr << "Could NOT complete stage 2.1." << std::endl;
00266     std::cerr << "Exiting..." << std::endl;
00267     return false;
00268 }
00269 #endif
00270
00271
00272 abort_construction = ComputePreliminaryApproximations();
00273
00274 #ifdef MTK_PERFORM_PREVENTIONS
00275 if (!abort_construction) {
00276     std::cerr << "Could NOT complete stage 2.2." << std::endl;
00277     std::cerr << "Exiting..." << std::endl;
00278     return false;
00279 }
00280 #endif
00281
00282
00283 abort_construction = ComputeWeights();
00284
00285 #ifdef MTK_PERFORM_PREVENTIONS
00286 if (!abort_construction) {
00287     std::cerr << "Could NOT complete stage 2.3." << std::endl;
00288     std::cerr << "Exiting..." << std::endl;
00289     return false;
00290 }
00291 #endif
00292
00293
00294 abort_construction = ComputeStencilBoundaryGrid();
00295
00296 #ifdef MTK_PERFORM_PREVENTIONS
00297 if (!abort_construction) {
00298     std::cerr << "Could NOT complete stage 2.4." << std::endl;
00299     std::cerr << "Exiting..." << std::endl;
00300     return false;
00301 }
00302 #endif
00303
00304
00305 } // End of: if (dim_null_ > 0);
00306
00307
00308
00309
00310
00311 // Once we have the following three collections of data:
00312 // (a) the coefficients for the interior,
00313 // (b) the coefficients for the boundary (if it applies),
00314 // (c) and the weights (if it applies),
00315 // we will store everything in the output array:
00316
00317 abort_construction = AssembleOperator();
00318
00319 #ifdef MTK_PERFORM_PREVENTIONS
00320 if (!abort_construction) {
00321     std::cerr << "Could NOT complete stage 3." << std::endl;
00322     std::cerr << "Exiting..." << std::endl;
00323     return false;
00324 }
00325 #endif
00326
00327 return true;
00328 }
00329
00330 int mtk::Div1D::num_bndy_coeffs() const {
00331
00332     return num_bndy_coeffs_;
00333 }
00334
00335 mtk::Real *mtk::Div1D::coeffs_interior() const {
00336
00337     return coeffs_interior_;
00338 }
```

```

00339
00340 mtk::Real *mtk::Div1D::weights_crs() const {
00341     return weights_crs_;
00342 }
00343 }
00344
00345 mtk::Real *mtk::Div1D::weights_cbs() const {
00346
00347     return weights_cbs_;
00348 }
00349
00350 int mtk::Div1D::num_feasible_sols() const {
00351
00352     return num_feasible_sols_;
00353 }
00354
00355 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00356
00357     mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00358
00359     auto counter = 0;
00360     for (auto ii = 0; ii < dim_null_; ++ii) {
00361         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00362             xx.SetValue(ii, jj, divergence_[2*order_accuracy_ + 1 + counter]);
00363             counter++;
00364         }
00365     }
00366
00367     return xx;
00368 }
00369
00370 std::vector<mtk::Real> mtk::Div1D::sums_rows_mim_bndy() const {
00371
00372     return sums_rows_mim_bndy_;
00373 }
00374
00375 mtk::Real mtk::Div1D::mimetic_measure() const {
00376
00377     return mimetic_measure_;
00378 }
00379
00380 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(
00381     const UniStgGrid1D &grid) const {
00382
00383     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00384
00385 #ifdef MTK_PERFORM_PREVENTIONS
00386     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00387     mtk::Tools::Prevent(nn < 3*order_accuracy_- 1, __FILE__, __LINE__, __func__);
00388     mtk::Tools::Prevent(grid.field_nature() !=
00389                         mtk::FieldNature::VECTOR,
00390                         __FILE__, __LINE__, __func__);
00391 #endif
00392     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00393
00394     int dd_num_rows = nn + 2;
00395     int dd_num_cols = nn + 1;
00396     int elements_per_row = num_bndy_coeffs_;
00397     int num_extra_rows = dim_null_;
00398
00399 // Output matrix featuring sizes for divergence operators.
00400     mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00401
00402     out.set_encoded_operator(mtk::EncodedOperator::DIVERGENCE
00403 );
00404
00405
00406     for (int ii = 0; ii < num_extra_rows; ++ii) {
00407         int ee{ii};
00408         for (int jj = 0; jj < elements_per_row; ++jj) {
00409             // We index at ii + 1 to secure a padded divergence matrix.
00410             out.SetValue(ii + 1, jj, mim_bndy_[ee]*inv_delta_x);
00411             ee += num_extra_rows;
00412         }
00413     }
00414
00415     for (auto ii = num_extra_rows + 1;
00416          ii < dd_num_rows - num_extra_rows - 1; ii++) {
00417         auto jj = ii - num_extra_rows - 1;

```

```

00420     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00421         out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00422     }
00423 }
00424
00425
00426 for (int ii = 0; ii < num_extra_rows; ++ii) {
00427     int ee{ii};
00428     for (int jj = 0; jj < elements_per_row; ++jj) {
00429         int rr{dd_num_rows - 2 - ii};
00430         int cc{dd_num_cols - 1 - jj};
00431         out.SetValue(rr, cc, -mim_bndy_[ee]*inv_delta_x);
00432         ee += num_extra_rows;
00433     }
00434 }
00435 }
00436
00437 return out;
00438 }
00439
00440 mtk::DenseMatrix mtk::Div1D::ReturnAsDimensionlessDenseMatrix
(
00441     int num_cells_x) const {
00442
00443     int nn{num_cells_x}; // Number of cells on the grid.
00444
00445 #ifdef MTK_PERFORM_PREVENTIONS
00446     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00447     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00448 #endif
00449
00450     int dd_num_rows = nn + 2;
00451     int dd_num_cols = nn + 1;
00452     int elements_per_row = num_bndy_coeffs_;
00453     int num_extra_rows = dim_null_;
00454
00455     // Output matrix featuring sizes for gradient operators.
00456     mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00457
00458     out.set_encoded_operator(mtk::EncodedOperator::DIVERGENCE
00459 );
00460
00461
00462     for (int ii = 0; ii < num_extra_rows; ++ii) {
00463         int ee{ii};
00464         for (int jj = 0; jj < elements_per_row; ++jj) {
00465             // We index at ii + 1 to secure a padded divergence matrix.
00466             out.SetValue(ii + 1, jj, mim_bndy_[ee]);
00467             ee += num_extra_rows;
00468         }
00469     }
00470
00471
00472     for (auto ii = num_extra_rows + 1;
00473          ii < dd_num_rows - num_extra_rows - 1; ii++) {
00474         auto jj = ii - num_extra_rows - 1;
00475         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00476             out.SetValue(ii, jj, coeffs_interior_[cc]);
00477         }
00478     }
00479 }
00480
00481
00482
00483     for (int ii = 0; ii < num_extra_rows; ++ii) {
00484         int ee{ii};
00485         for (int jj = 0; jj < elements_per_row; ++jj) {
00486             int rr{dd_num_rows - 2 - ii};
00487             int cc{dd_num_cols - 1 - jj};
00488             out.SetValue(rr, cc, -mim_bndy_[ee]);
00489             ee += num_extra_rows;
00490         }
00491     }
00492
00493 return out;
00494 }
00495
00496 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00497
00498
00499     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00500
00501     try {
00502         pp = new mtk::Real[order_accuracy_];
00503     }

```

```

00504     } catch (std::bad_alloc &memory_allocation_exception) {
00505         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00506             std::endl;
00507         std::cerr << memory_allocation_exception.what() << std::endl;
00508     }
00509     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00510
00511 #ifdef MTK_PRECISION_DOUBLE
00512     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00513 #else
00514     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00515 #endif
00516
00517     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00518         pp[ii] = pp[ii - 1] + mtk::kOne;
00519     }
00520
00521 #if MTK_VERBOSE_LEVEL > 3
00522     std::cout << "pp =" << std::endl;
00523     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00524         std::cout << std::setw(12) << pp[ii];
00525     }
00526     std::cout << std::endl << std::endl;
00527 #endif
00528
00529
00530     bool transpose{false};
00531
00532     mtk::DenseMatrix vander_matrix(pp,
00533                                     order_accuracy_,
00534                                     order_accuracy_,
00535                                     transpose);
00536
00537
00538 #if MTK_VERBOSE_LEVEL > 4
00539     std::cout << "vander_matrix = " << std::endl;
00540     std::cout << vander_matrix << std::endl;
00541 #endif
00542
00543
00544     try {
00545         coeffs_interior_ = new mtk::Real[order_accuracy_];
00546     } catch (std::bad_alloc &memory_allocation_exception) {
00547         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00548             std::endl;
00549         std::cerr << memory_allocation_exception.what() << std::endl;
00550     }
00551     memset(coeffs_interior_,
00552            mtk::kZero,
00553            sizeof(coeffs_interior_[0])*order_accuracy_);
00554
00555     coeffs_interior_[1] = mtk::kOne;
00556
00557
00558 #if MTK_VERBOSE_LEVEL > 3
00559     std::cout << "oo =" << std::endl;
00560     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00561         std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00562     }
00563     std::cout << std::endl;
00564 #endif
00565
00566
00567     int info=mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00568                                                     coeffs_interior_);
00569
00570
00571 #ifdef MTK_PERFORM_PREVENTIONS
00572     if (!info) {
00573         std::cout << "System solved! Interior stencil attained!" << std::endl;
00574         std::cout << std::endl;
00575     }
00576     else {
00577         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00578         std::cerr << "Exiting..." << std::endl;
00579         return false;
00580     }
00581 #endif
00582
00583 #if MTK_VERBOSE_LEVEL > 3
00584     std::cout << "coeffs_interior_ =" << std::endl;
00585     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00586         std::cout << std::setw(12) << coeffs_interior_[ii];
00587     }
00588 
```

```

00588     std::cout << std::endl << std::endl;
00589     #endif
00590
00591     delete [] pp;
00592     pp = nullptr;
00593
00594     return true;
00595 }
00596
00597 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
00598
00599     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00600
00601
00602     try {
00603         gg = new mtk::Real[num_bndy_coeffs_];
00604     } catch (std::bad_alloc &memory_allocation_exception) {
00605         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00606         std::endl;
00607         std::cerr << memory_allocation_exception.what() << std::endl;
00608     }
00609     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00610
00611     #ifdef MTK_PRECISION_DOUBLE
00612     gg[0] = -1.0/2.0;
00613     #else
00614     gg[0] = -1.0f/2.0f;
00615     #endif
00616
00617     for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00618         gg[ii] = gg[ii - 1] + mtk::kOne;
00619     }
00620
00621     #if MTK_VERBOSE_LEVEL > 3
00622     std::cout << "gg =" << std::endl;
00623     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00624         std::cout << std::setw(12) << gg[ii];
00625     }
00626     std::cout << std::endl << std::endl;
00627     #endif
00628
00629
00630
00631     bool tran{true}; // Should I transpose the Vandermonde matrix.
00632
00633     mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00634
00635     #if MTK_VERBOSE_LEVEL > 4
00636     std::cout << "vv_west_t =" << std::endl;
00637     std::cout << vv_west_t << std::endl;
00638     #endif
00639
00640
00641     mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00642     (vv_west_t));
00643
00644     #if MTK_VERBOSE_LEVEL > 4
00645     std::cout << "QQ^T = " << std::endl;
00646     std::cout << qq_t << std::endl;
00647     #endif
00648
00649
00650
00651     int KK_num_rows_{num_bndy_coeffs_};
00652     int KK_num_cols_{dim_null_};
00653
00654     mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00655
00656     for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00657         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00658             KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00659                 qq_t.data()[ii*num_bndy_coeffs_ + jj];
00660         }
00661     }
00662
00663     #if MTK_VERBOSE_LEVEL > 2
00664     std::cout << "KK =" << std::endl;
00665     std::cout << KK << std::endl;
00666     std::cout << "KK.num_rows() = " << KK.num_rows() << std::endl;
00667     std::cout << "KK.num_cols() = " << KK.num_cols() << std::endl;
00668     std::cout << std::endl;
00669     #endif
00670
00671
00672

```

```

00673 // Scale thus requesting that the last entries of the attained basis for the
00674 // null-space, adopt the pattern we require.
00675 // Essentially we will implement the following MATLAB pseudo-code:
00676 // scalers = KK(num_bndy_approx - (dim_null - 1):num_bndy_approx,:)\B
00677 // SK = KK*scalers
00678 // where SK is the scaled null-space.
00679
00680 // In this point, we almost have all the data we need correctly allocated
00681 // in memory. We will create the matrix II_, and elements we wish to scale in
00682 // the KK array. Using the concept of the leading dimension, we could just
00683 // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00684 // GET how does it work. So I will just create a matrix with the content of
00685 // this array that we need, solve for the scalers and then scale the
00686 // whole KK:
00687
00688 // We will then create memory for that sub-matrix of KK (SUBK).
00689
00690 mtk::DenseMatrix SUBK(dim_null_, dim_null_);
00691
00692 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00693     for (auto jj = 0; jj < dim_null_; ++jj) {
00694         SUBK.data()[(ii - (num_bndy_coeffs_ - dim_null_))*dim_null_ + jj] =
00695             KK.data()[ii*dim_null_ + jj];
00696     }
00697 }
00698
00699 #if MTK_VERBOSE_LEVEL > 4
00700 std::cout << "SUBK =" << std::endl;
00701 std::cout << SUBK << std::endl;
00702 #endif
00703
00704 SUBK.Transpose();
00705
00706 #if MTK_VERBOSE_LEVEL > 4
00707 std::cout << "SUBK^T =" << std::endl;
00708 std::cout << SUBK << std::endl;
00709 #endif
00710
00711 bool padded{false};
00712 tran = false;
00713
00714 mtk::DenseMatrix II(dim_null_, padded, tran);
00715
00716 #if MTK_VERBOSE_LEVEL > 4
00717 std::cout << "II =" << std::endl;
00718 std::cout << II << std::endl;
00719 #endif
00720
00721 // Solve the system to compute the scalers.
00722 // An example of the system to solve, for k = 8, is:
00723 //
00724 // SUBK*scalers = II_ or
00725 //
00726 // | 0.386018 -0.0339244 -0.129478 |      | 1 0 0 |
00727 // | -0.119774 0.0199423 0.0558632 | *scalers = | 0 1 0 |
00728 // | 0.0155708 -0.00349546 -0.00853182 |      | 0 0 1 |
00729 //
00730 // Notice this is a nrhs = 3 system.
00731 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00732 // will be stored in the created identity matrix.
00733 // Let us first transpose SUBK (because of LAPACK):
00734
00735 int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00736
00737 #ifdef MTK_PERFORM_PREVENTIONS
00738 if (!info) {
00739     std::cout << "System successfully solved!" <<
00740         std::endl;
00741 } else {
00742     std::cerr << "Something went wrong solving system! info = " << info <<
00743         std::endl;
00744     std::cerr << "Exiting..." << std::endl;
00745     return false;
00746 }
00747 std::cout << std::endl;
00748 #endif
00749
00750 #if MTK_VERBOSE_LEVEL > 4
00751 std::cout << "Computed scalers:" << std::endl;
00752 std::cout << II << std::endl;
00753 #endif

```

```

00754
00755 // Multiply the two matrices to attain a scaled basis for null-space.
00756
00757 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00758
00759 #if MTK_VERBOSE_LEVEL > 4
00760 std::cout << "Rational basis for the null-space:" << std::endl;
00761 std::cout << rat_basis_null_space_ << std::endl;
00762 #endif
00763
00764 // At this point, we have a rational basis for the null-space, with the
00765 // pattern we need! :)
00766
00767 delete [] gg;
00768 gg = nullptr;
00769
00770 return true;
00771 }
00772
00773 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00774
00775 mtk::Real *gg{}; // Generator vector for the first approximation.
00776
00777 try {
00778     gg = new mtk::Real[num_bndy_coeffs_];
00779 } catch (std::bad_alloc &memory_allocation_exception) {
00780     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00781 std::endl;
00782     std::cerr << memory_allocation_exception.what() << std::endl;
00783 }
00784 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00785
00786 #ifdef MTK_PRECISION_DOUBLE
00787 gg[0] = -1.0/2.0;
00788 #else
00789 gg[0] = -1.0f/2.0f;
00790 #endif
00791 for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00792     gg[ii] = gg[ii - 1] + mtk::kOne;
00793 }
00794
00795 #if MTK_VERBOSE_LEVEL > 3
00796 std::cout << "gg0 =" << std::endl;
00797 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00798     std::cout << std::setw(12) << gg[ii];
00799 }
00800 std::cout << std::endl << std::endl;
00801 #endif
00802
00803 // Allocate 2D array to store the collection of preliminary approximations.
00804
00805 try {
00806     prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00807 } catch (std::bad_alloc &memory_allocation_exception) {
00808     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00809 std::endl;
00810     std::cerr << memory_allocation_exception.what() << std::endl;
00811 }
00812
00813 memset(prem_apps_,
00814         mtk::kZero,
00815         sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00816
00817
00818 for (auto ll = 0; ll < dim_null_; ++ll) {
00819
00820     // Re-check new generator vector for every iteration except for the first.
00821     #if MTK_VERBOSE_LEVEL > 3
00822         if (ll > 0) {
00823             std::cout << "gg" << ll << " = " << std::endl;
00824             for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00825                 std::cout << std::setw(12) << gg[ii];
00826             }
00827             std::cout << std::endl << std::endl;
00828         }
00829     #endif
00830
00831     bool transpose{false};
00832
00833     mtk::DenseMatrix AA_(gg,
00834                           num_bndy_coeffs_, order_accuracy_ + 1,
00835
00836
00837

```

```

00838                     transpose);
00839
00840 #if MTK_VERBOSE_LEVEL > 4
00841 std::cout << "AA_" << ll << " = " << std::endl;
00842 std::cout << AA_ << std::endl;
00843 #endif
00844
00845
00846
00847     mtk::Real *ob{};
00848
00849     auto ob_ld = num_bndy_coeffs_;
00850
00851     try {
00852         ob = new mtk::Real[ob_ld];
00853     } catch (std::bad_alloc &memory_allocation_exception) {
00854         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00855             std::endl;
00856         std::cerr << memory_allocation_exception.what() << std::endl;
00857     }
00858     memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00859
00860     ob[1] = mtk::kOne;
00861
00862 #if MTK_VERBOSE_LEVEL > 4
00863     std::cout << "ob = " << std::endl << std::endl;
00864     for (auto ii = 0; ii < ob_ld; ++ii) {
00865         std::cout << std::setw(12) << ob[ii] << std::endl;
00866     }
00867     std::cout << std::endl;
00868 #endif
00869
00870
00871 // However, this is an under-determined system of equations. So we can not
00872 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00873 // our LAPACKAdapter class.
00874
00875
00876     int info_{
00877         mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00878         ob, ob_ld)};
00879
00880 #ifdef MTK_PERFORM_PREVENTIONS
00881     if (!info_) {
00882         std::cout << "System successfully solved!" << std::endl << std::endl;
00883     } else {
00884         std::cerr << "Error solving system! info = " << info_ << std::endl;
00885     }
00886 #endif
00887
00888 #if MTK_VERBOSE_LEVEL > 3
00889     std::cout << "ob =" << std::endl;
00890     for (auto ii = 0; ii < ob_ld; ++ii) {
00891         std::cout << std::setw(12) << ob[ii] << std::endl;
00892     }
00893     std::cout << std::endl;
00894 #endif
00895
00896
00897 // This implies a DAXPY operation. However, we must construct the arguments
00898 // for this operation.
00899
00900 // Save them into the ob_bottom array:
00901
00902     Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00903
00904
00905     try {
00906         ob_bottom = new mtk::Real[dim_null_];
00907     } catch (std::bad_alloc &memory_allocation_exception) {
00908         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00909             std::endl;
00910         std::cerr << memory_allocation_exception.what() << std::endl;
00911     }
00912     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00913
00914     for (auto ii = 0; ii < dim_null_; ++ii) {
00915         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00916     }
00917
00918 #if MTK_VERBOSE_LEVEL > 3
00919     std::cout << "ob_bottom =" << std::endl;
00920     for (auto ii = 0; ii < dim_null_; ++ii) {
00921         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;

```

```

00922     }
00923     std::cout << std::endl;
00924 #endif
00925
00927
00928 // We must computed an scaled ob, sob, using the scaled null-space in
00929 // rat_basis_null_space_.
00930 // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00931 // or:                  ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00932 // thus:                 Y =      a*A      *x      +      b*Y (DAXPY).
00933
00934 #if MTK_VERBOSE_LEVEL > 3
00935     std::cout << "Rational basis for the null-space:" << std::endl;
00936     std::cout << rat_basis_null_space_ << std::endl;
00937 #endif
00938
00939 mtk::Real alpha{-mtk::kOne};
00940 mtk::Real beta{mtk::kOne};
00941
00942 mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00943                                 ob_bottom, beta, ob);
00944
00945 #if MTK_VERBOSE_LEVEL > 3
00946     std::cout << "scaled ob:" << std::endl;
00947     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00948         std::cout << std::setw(12) << ob[ii] << std::endl;
00949     }
00950     std::cout << std::endl;
00951 #endif
00952
00953 // We save the recently scaled solution, into an array containing these.
00954 // We can NOT start building the pi matrix, simply because I want that part
00955 // to be separated since its construction depends on the algorithm we want
00956 // to implement.
00957
00958 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00959     prem_apps_[ii*dim_null_ + ll] = ob[ii];
00960 }
00961
00962 // After the first iteration, simply shift the entries of the last
00963 // generator vector used:
00964 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00965     gg[ii]--;
00966 }
00967
00968 // Garbage collection for this loop:
00969 delete[] ob;
00970 ob = nullptr;
00971
00972 delete[] ob_bottom;
00973 ob_bottom = nullptr;
00974 } // End of: for (ll = 0; ll < dim_null; ll++);
00975
00976 #if MTK_VERBOSE_LEVEL > 4
00977 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00978 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00979     for (auto jj = 0; jj < dim_null_; ++jj) {
00980         std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00981     }
00982     std::cout << std::endl;
00983 }
00984 std::cout << std::endl;
00985 #endif
00986
00987 delete[] gg;
00988 gg = nullptr;
00989
00990 return true;
00991 }
00992
00993 bool mtk::Div1D::ComputeWeights(void) {
00994
00995 // Matrix to compute the weights as in the CRSA.
00996 mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00997
00998 // Assemble the pi matrix using:
01000 // 1. The collection of scaled preliminary approximations.
01001 // 2. The collection of coefficients approximating at the interior.
01002 // 3. The scaled basis for the null-space.
01003
01004

```

```

01005 // 1.1. Process array of scaled preliminary approximations.
01006
01007 // These are queued in scaled_solutions. Each one of these, will be a column
01008 // of the pi matrix:
01009 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01010     for (auto jj = 0; jj < dim_null_; ++jj) {
01011         pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01012             prem_apps_[ii*dim_null_ + jj];
01013     }
01014 }
01015
01016 // 1.2. Add columns from known stencil approximating at the interior.
01017
01018 // However, these must be padded by zeros, according to their position in the
01019 // final pi matrix:
01020 auto mm = 0;
01021 for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
01022     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01023         pi.data()[(ii + mm)*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01024             coeffs_interior_[ii];
01025     }
01026     ++mm;
01027 }
01028
01029 rat_basis_null_space_.OrderColMajor();
01030
01031 #if MTK_VERBOSE_LEVEL > 4
01032 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01033 std::cout << rat_basis_null_space_ << std::endl;
01034 #endif
01035
01036 // 1.3. Add final set of columns: rational basis for null-space.
01037
01038 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01039     jj < num_bndy_coeffs_ - 1;
01040     ++jj) {
01041     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01042         auto og =
01043             (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01044         auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01045         pi.data()[de] = rat_basis_null_space_.data()[og];
01046     }
01047 }
01048
01049 #if MTK_VERBOSE_LEVEL > 3
01050 std::cout << "coeffs_interior_ =" << std::endl;
01051 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01052     std::cout << std::setw(12) << coeffs_interior_[ii];
01053 }
01054 std::cout << std::endl << std::endl;
01055 #endif
01056
01057 #if MTK_VERBOSE_LEVEL > 4
01058 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01059 std::cout << pi << std::endl;
01060 #endif
01061
01062
01063 // This imposes the mimetic condition.
01064
01065 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01066
01067 try {
01068     hh = new mtk::Real[num_bndy_coeffs_];
01069 } catch (std::bad_alloc &memory_allocation_exception) {
01070     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01071     std::endl;
01072     std::cerr << memory_allocation_exception.what() << std::endl;
01073 }
01074
01075 memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01076
01077 hh[0] = -mtk::kOne;
01078 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01079     auto aux_xx = mtk::kZero;
01080     for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01081         aux_xx += coeffs_interior_[jj];
01082     }
01083     hh[ii] = -mtk::kOne*aux_xx;
01084 }
01085
01086
01087

```

```

01088 // That is, we construct a system, to solve for the weights.
01089
01090 // Once again we face the challenge of solving with LAPACK. However, for the
01091 // CRSAs, this matrix PI is over-determined, since it has more rows than
01092 // unknowns. However, according to the theory, the solution to this system is
01093 // unique. We will use dgels_.
01094
01095 try {
01096     weights_cbs_ = new mtk::Real[num_bndy_coefffs_];
01097 } catch (std::bad_alloc &memory_allocation_exception) {
01098     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01099     std::endl;
01100     std::cerr << memory_allocation_exception.what() << std::endl;
01101 }
01102 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coefffs_);
01103
01104 int weights_ld{pi.num_cols() + 1};
01105
01106 // Preserve hh.
01107 std::copy(hh, hh + weights_ld, weights_cbs_);
01108
01109 pi.Transpose();
01110
01111 int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
01112     pi,
01113                                         weights_cbs_,
01114                                         weights_ld)};
01115 #ifdef MTK_PERFORM_PREVENTIONS
01116 if (!info) {
01117     std::cout << "System successfully solved!" << std::endl << std::endl;
01118 } else {
01119     std::cerr << "Error solving system! info = " << info << std::endl;
01120 }
01121 #endif
01122
01123 #if MTK_VERBOSE_LEVEL > 3
01124 std::cout << "hh =" << std::endl;
01125 for (auto ii = 0; ii < num_bndy_coefffs_; ++ii) {
01126     std::cout << std::setw(11) << hh[ii] << std::endl;
01127 }
01128 std::cout << std::endl;
01129 #endif
01130
01131 // Preserve the original weights for research.
01132
01133 try {
01134     weights_crs_ = new mtk::Real[num_bndy_coefffs_];
01135 } catch (std::bad_alloc &memory_allocation_exception) {
01136     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01137     std::endl;
01138     std::cerr << memory_allocation_exception.what() << std::endl;
01139 }
01140 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coefffs_);
01141
01142 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01143
01144 #if MTK_VERBOSE_LEVEL > 3
01145 std::cout << "weights_CRSA + lambda =" << std::endl;
01146 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01147     std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01148 }
01149 std::cout << std::endl;
01150 #endif
01151
01152
01153 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01154
01155     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01156
01157     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01158         for (auto jj = 0; jj < dim_null_; ++jj) {
01159             phi.data()[ii*(order_accuracy_) + jj] = prem_apps_[ii*dim_null_ + jj];
01160         }
01161     }
01162
01163     int aux{}; // Auxiliary variable.
01164     for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01165         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01166             phi.data()[(ii + aux)*order_accuracy_ + jj] = coeffs_interior_[ii];
01167         }
01168     }
01169 }
```

```

01170         }
01171         ++aux;
01172     }
01173
01174     for(auto jj=order_accuracy_- 1; jj >=order_accuracy_- dim_null_; jj--) {
01175         for(auto ii=0; ii<order_accuracy_- + 1; ++ii) {
01176             phi.data()[ii*order_accuracy_-+jj] = mtk::kZero;
01177         }
01178     }
01179
01180     for (auto jj = 0; jj < order_accuracy_- + 1; ++jj) {
01181         for (auto ii = 0; ii < dim_null_-; ++ii) {
01182             phi.data()[(ii + order_accuracy_- - dim_null_- + jj*order_accuracy_-)] =
01183                 -prem_apps_[(dim_null_- - ii - 1 + jj*dim_null_-)];
01184         }
01185     }
01186
01187     for(auto ii = 0; ii < order_accuracy_-/2; ++ii) {
01188         for (auto jj = dim_null_- + 2; jj < order_accuracy_-; ++jj) {
01189             auto swap = phi.data()[ii*order_accuracy_-+jj];
01190             phi.data()[ii*order_accuracy_- + jj] =
01191                 phi.data()[(order_accuracy_-ii)*order_accuracy_-+jj];
01192             phi.data()[(order_accuracy_-ii)*order_accuracy_-+jj] = swap;
01193         }
01194     }
01195
01196 #if MTK_VERBOSE_LEVEL > 4
01197 std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01198 std::cout << phi << std::endl;
01199 #endif
01200
01201
01202     mtk::Real *lamed{}; // Used to build big lambda.
01203
01204
01205     try {
01206         lamed = new mtk::Real[dim_null_-];
01207     } catch (std::bad_alloc &memory_allocation_exception) {
01208         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01209             std::endl;
01210         std::cerr << memory_allocation_exception.what() << std::endl;
01211     }
01212     memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01213
01214     for (auto ii = 0; ii < dim_null_-; ++ii) {
01215         lamed[ii] = hh[ii + order_accuracy_- + 1];
01216     }
01217
01218 #if MTK_VERBOSE_LEVEL > 3
01219 std::cout << "lamed =" << std::endl;
01220 for (auto ii = 0; ii < dim_null_-; ++ii) {
01221     std::cout << std::setw(12) << lamed[ii] << std::endl;
01222 }
01223 std::cout << std::endl;
01224 #endif
01225
01226     for (auto ii = 0; ii < num_bndy_coeffs_-; ++ii) {
01227         mtk::Real temp = mtk::kZero;
01228         for(auto jj = 0; jj < dim_null_-; ++jj) {
01229             temp = temp +
01230                 lamed[jj]*rat_basis_null_space_.data()[(jj*num_bndy_coeffs_- + ii)];
01231         }
01232         hh[ii] = hh[ii] - temp;
01233     }
01234
01235 #if MTK_VERBOSE_LEVEL > 3
01236 std::cout << "big_lambda =" << std::endl;
01237 for (auto ii = 0; ii < num_bndy_coeffs_-; ++ii) {
01238     std::cout << std::setw(12) << hh[ii] << std::endl;
01239 }
01240 std::cout << std::endl;
01241 #endif
01242
01243 #ifdef MTK_VERBOSE_WEIGHTS
01244 int copy_result{1};
01245 #else
01246 int copy_result{};
01247 #endif
01248
01249     mtk::Real normerr_; // Norm of the error for the solution on each row.
01250
01251
01252

```

```

01253     int minrow_{std::numeric_limits<int>::infinity()};
01254
01255     mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_crs_,
01256     order_accuracy_)};
01257     mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01258
01259 #ifdef MTK_VERBOSE_WEIGHTS
01260     std::ofstream table("div_1d_" + std::to_string(order_accuracy_) +
01261     "_weights.tex");
01262
01263     table << "\\begin{tabular}[c]{c}";
01264     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01265         table << 'c';
01266     }
01267     table << ":c}\\n\\toprule\\nRow & ";
01268     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01269         table << "$ q_{ " + std::to_string(ii) + " }$ &";
01270     }
01271     table << " Relative error \\\\\\n\\midrule\\n";
01272 #endif
01273
01274     for(auto row_= 0; row_ < order_accuracy_ + 1; ++row_) {
01275         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01276         data(),
01277                                         order_accuracy_ + 1,
01278                                         order_accuracy_,
01279                                         order_accuracy_,
01280                                         hh,
01281                                         weights_cbs_,
01282                                         row_,
01283                                         mimetic_threshold_,
01284                                         copy_result);
01285
01286     mtk::Real aux{normerr_/norm_};
01287
01288 #if MTK_VERBOSE_LEVEL > 2
01289     std::cout << "Relative norm: " << aux << " " << std::endl;
01290     std::cout << std::endl;
01291 #endif
01292
01293     num_feasible_sols_ = num_feasible_sols_ +
01294     (int) (normerr_ != std::numeric_limits<mtk::Real>::infinity());
01295
01296     if (aux < minnorm_) {
01297         minnorm_ = aux;
01298         minrow_= row_;
01299     }
01300
01301 #ifdef MTK_VERBOSE_WEIGHTS
01302     table << std::to_string(row_ + 1) << " & ";
01303     if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01304         for (int ii = 1; ii <= order_accuracy_; ++ii) {
01305             table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01306         }
01307         table << std::to_string(aux) << " \\\\\\" << std::endl;
01308     } else {
01309         table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
01310         "}{c}{\$\\emptyset\$} & ";
01311         table << " - \\\\\\" << std::endl;
01312     }
01313 #endif
01314
01315 #ifdef MTK_VERBOSE_WEIGHTS
01316     table << "\\midrule" << std::endl;
01317     table << "CRS weights:";
01318     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01319         table << " & " << std::to_string(weights_crs_[ii - 1]);
01320     }
01321     table << " & - \\\\\\n\\bottomrule\\n\\end{tabular}" << std::endl;
01322     table.close();
01323 #endif
01324
01325 #if MTK_VERBOSE_LEVEL > 3
01326     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01327     std::endl;
01328     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01329         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01330     }
01331     std::cout << std::endl;
01332 #endif

```

```

01333
01334     // After we know which row yields the smallest relative norm that row is
01335     // chosen to be the objective function and the result of the optimizer is
01336     // chosen to be the new weights_.
01337
01338     #if MTK_VERBOSE_LEVEL > 2
01339         std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01340             minrow_ + 1 << std::endl;
01341         std::cout << std::endl;
01342     #endif
01343
01344     copy_result = 1;
01345     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01346         data(),
01347                                         order_accuracy_ + 1,
01348                                         order_accuracy_,
01349                                         order_accuracy_,
01350                                         hh,
01351                                         weights_cbs_,
01352                                         minrow_,
01353                                         mimetic_threshold_,
01354                                         copy_result);
01355     mtk::Real aux_(normerr_/norm_);
01356     #if MTK_VERBOSE_LEVEL > 2
01357         std::cout << "Relative norm: " << aux_ << std::endl;
01358         std::cout << std::endl;
01359     #endif
01360     delete [] lamed;
01361     lamed = nullptr;
01362 }
01363
01364 delete [] hh;
01365 hh = nullptr;
01366
01367 return true;
01368 }
01369
01370 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01371
01372     #if MTK_VERBOSE_LEVEL > 3
01373         std::cout << "weights_CBSA + lambda =" << std::endl;
01374         for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01375             std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01376         }
01377         std::cout << std::endl;
01378     #endif
01379
01380     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01381
01382     try {
01383         lambda = new mtk::Real[dim_null_];
01384     } catch (std::bad_alloc &memory_allocation_exception) {
01385         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01386             std::endl;
01387         std::cerr << memory_allocation_exception.what() << std::endl;
01388     }
01389     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01390
01391     for (auto ii = 0; ii < dim_null_; ++ii) {
01392         lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01393     }
01394
01395     #if MTK_VERBOSE_LEVEL > 3
01396         std::cout << "lambda =" << std::endl;
01397         for (auto ii = 0; ii < dim_null_; ++ii) {
01398             std::cout << std::setw(12) << lambda[ii] << std::endl;
01399         }
01400         std::cout << std::endl;
01401     #endif
01402
01403     mtk::Real *alpha{}; // Collection of alpha values.
01404
01405     try {
01406         alpha = new mtk::Real[dim_null_];
01407     } catch (std::bad_alloc &memory_allocation_exception) {
01408         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01409             std::endl;
01410         std::cerr << memory_allocation_exception.what() << std::endl;
01411
01412
01413
01414

```

```

01415     }
01416     memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01417
01418     for (auto ii = 0; ii < dim_null_; ++ii) {
01419         alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01420     }
01421
01422 #if MTK_VERBOSE_LEVEL > 3
01423     std::cout << "alpha =" << std::endl;
01424     for (auto ii = 0; ii < dim_null_; ++ii) {
01425         std::cout << std::setw(12) << alpha[ii] << std::endl;
01426     }
01427     std::cout << std::endl;
01428 #endif
01429
01430
01431     try {
01432         mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01433     } catch (std::bad_alloc &memory_allocation_exception) {
01434         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01435             std::endl;
01436         std::cerr << memory_allocation_exception.what() << std::endl;
01437     }
01438     memset(mim_bndy_,
01439            mtk::kZero,
01440            sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01441
01442     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01443         for (auto jj = 0; jj < dim_null_; ++jj) {
01444             mim_bndy_[ii*dim_null_ + jj] =
01445                 prem_apps_[ii*dim_null_ + jj] +
01446                 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01447         }
01448     }
01449
01450 #if MTK_VERBOSE_LEVEL > 3
01451     std::cout << "Collection of mimetic approximations:" << std::endl;
01452     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01453         for (auto jj = 0; jj < dim_null_; ++jj) {
01454             std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01455         }
01456         std::cout << std::endl;
01457     }
01458 }
01459     std::cout << std::endl;
01460 #endif
01461
01462
01463     for (auto ii = 0; ii < dim_null_; ++ii) {
01464         sums_rows_mim_bndy_.push_back(mtk::kZero);
01465         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01466             sums_rows_mim_bndy_[ii] += mim_bndy_[jj*dim_null_ + ii];
01467         }
01468     }
01469
01470     mimetic_measure_ = *std::max_element(sums_rows_mim_bndy_.begin(),
01471                                         sums_rows_mim_bndy_.end());
01472
01473 #if MTK_VERBOSE_LEVEL > 3
01474     std::cout << "Row-wise sum of mimetic approximations:" << std::endl;
01475     for (auto ii = 0; ii < dim_null_; ++ii) {
01476         std::cout << std::setw(13) << sums_rows_mim_bndy_[ii];
01477     }
01478     std::cout << std::endl;
01479     std::cout << std::endl;
01480 #endif
01481
01482     delete[] lambda;
01483     lambda = nullptr;
01484
01485     delete[] alpha;
01486     alpha = nullptr;
01487
01488     return true;
01489 }
01490
01491
01492 bool mtk::Div1D::AssembleOperator(void) {
01493
01494 // The output array will have this form:
01495 // 1. The first entry of the array will contain used order order_accuracy_.
01496 // 2. The second entry of the array will contain the collection of
01497 // approximating coefficients for the interior of the grid.

```

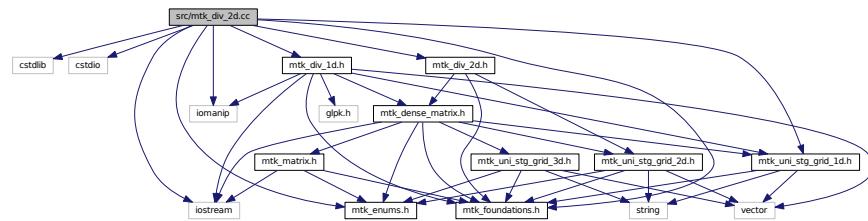
```

01498 // 3. IF order_accuracy_ > 2, then the third entry will contain a collection
01499 // of weights.
01500 // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the
01501 // collections of approximating coefficients for the west boundary of the
01502 // grid.
01503
01504 if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01505     divergence_length_ =
01506         1 + order_accuracy_ + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01507 } else {
01508     divergence_length_ = 1 + order_accuracy_;
01509 }
01510
01511 #if MTK_VERBOSE_LEVEL > 2
01512 std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01513 std::cout << std::endl;
01514 #endif
01515
01516 try {
01517     divergence_ = new double[divergence_length_];
01518 } catch (std::bad_alloc &memory_allocation_exception) {
01519     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01520     std::endl;
01521     std::cerr << memory_allocation_exception.what() << std::endl;
01522 }
01523 memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01524
01525 divergence_[0] = order_accuracy_;
01526
01527 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01528     divergence_[ii + 1] = coeffs_interior_[ii];
01529 }
01530
01531 if (order_accuracy_ > 2) {
01532     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01533         divergence_[(1 + order_accuracy_) + ii] = weights_cbs_[ii];
01534     }
01535 }
01536
01537 if (order_accuracy_ > 2) {
01538     auto offset = (2*order_accuracy_ + 1);
01539     int mm{};
01540     for (auto ii = 0; ii < dim_null_; ++ii) {
01541         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01542             divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01543             ++mm;
01544         }
01545     }
01546 }
01547 #if MTK_VERBOSE_LEVEL > 1
01548 std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01549 std::cout << std::endl;
01550 #endif
01551
01552 return true;
01553 }
```

18.97 src/mtk_div_2d.cc File Reference

Implements the class Div2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtkEnums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"
Include dependency graph for mtk_div_2d.cc:
```



18.97.1 Detailed Description

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

Author

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

Definition in file [mtk_div_2d.cc](#).

18.98 mtk_div_2d.cc

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
```

```
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtkEnums.h"
00065 #include "mtkUniStgGrid1D.h"
00066 #include "mtkDiv1D.h"
00067 #include "mtkDiv2D.h"
00068
00069 mtk::Div2D::Div2D():
00070     order_accuracy_(),
00071     mimetic_threshold_() {}
00072
00073 mtk::Div2D::Div2D(const Div2D &div):
00074     order_accuracy_(div.order_accuracy_),
00075     mimetic_threshold_(div.mimetic_threshold_) {}
00076
00077 mtk::Div2D::~Div2D() {}
00078
00079 bool mtk::Div2D::ConstructDiv2D(const
00080                                     mtk::UniStgGrid2D &grid,
00081                                     int order_accuracy,
00082                                     mtk::Real mimetic_threshold) {
00083
00084     int num_cells_x = grid.num_cells_x();
00085     int num_cells_y = grid.num_cells_y();
00086
00087     int mx = num_cells_x + 2; // Dx vertical dimension.
00088     int nx = num_cells_x + 1; // Dx horizontal dimension.
00089     int my = num_cells_y + 2; // Dy vertical dimension.
00090     int ny = num_cells_y + 1; // Dy horizontal dimension.
00091
00092     mtk::Div1D div;
00093
00094     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00095
00096     #ifdef MTK_PERFORM_PREVENTIONS
00097     if (!info) {
00098         std::cerr << "Mimetic div could not be built." << std::endl;
00099         return info;
00100     }
00101     #endif
00102
00103     auto west = grid.west_bndy();
00104     auto east = grid.east_bndy();
00105     auto south = grid.south_bndy();
00106     auto north = grid.east_bndy();
00107
00108     mtk::UniStgGrid1D grid_x(west, east, num_cells_x,
00109                             mtk::FieldNature::VECTOR);
00110     mtk::UniStgGrid1D grid_y(south, north, num_cells_y,
00111                             mtk::FieldNature::VECTOR);
00112
00113     bool padded{true};
```

```

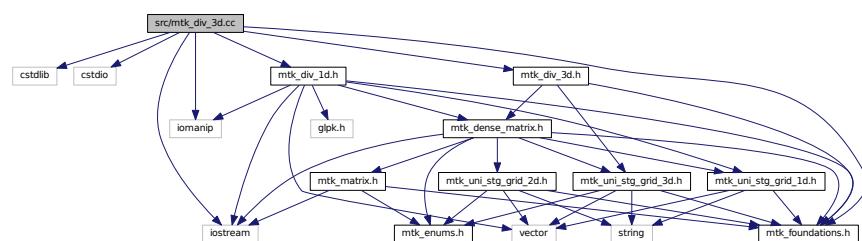
00114     bool transpose{false};
00115
00116     mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00117     mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00118
00119     mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00120     mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00121
00122 #if MTK_VERBOSE_LEVEL > 2
00123 std::cout << "Dx: " << mx << " by " << nx << std::endl;
00124 std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;
00125 std::cout << "Dy: " << my << " by " << ny << std::endl;
00126 std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00127 std::cout << "Div 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00128     nx*ny << std::endl;
00129#endif
00130
00131     mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00132
00133     for (auto ii = 0; ii < mx*my; ii++) {
00134         for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00135             d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00136         }
00137         for (auto kk = 0; kk < ny*num_cells_x; kk++) {
00138             d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00139         }
00140     }
00141
00142     divergence_ = d2d;
00143
00144     return info;
00145 }
00146
00147 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() const {
00148
00149     return divergence_;
00150 }
```

18.99 src/mtk_div_3d.cc File Reference

Implements the class Div3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtk_div_1d.h"
#include "mtk_div_3d.h"

Include dependency graph for mtk_div_3d.cc:
```



18.99.1 Detailed Description

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

Author

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

Definition in file [mtk_div_3d.cc](#).

18.100 mtk_div_3d.cc

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtk_div_1d.h"
00065 #include "mtk_div_3d.h"
00066
00067 mtk::Div3D::Div3D():
00068     order_accuracy_(),
00069     mimetic_threshold_() {}

```

```

00070
00071     mtk::Div3D::Div3D(const Div3D &grad):
00072         order_accuracy_(grad.order_accuracy_),
00073         mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075     mtk::Div3D::~Div3D() {}
00076
00077     bool mtk::Div3D::ConstructDiv3D(const
00078         mtk::UniStgGrid3D &grid,
00079             int order_accuracy,
00080             mtk::Real mimetic_threshold) {
00081
00082         int num_cells_x = grid.num_cells_x();
00083         int num_cells_y = grid.num_cells_y();
00084         int num_cells_z = grid.num_cells_z();
00085
00086         int mx = num_cells_x + 1; // Dx vertical dimension.
00087         int nx = num_cells_x + 2; // Dx horizontal dimension.
00088         int my = num_cells_y + 1; // Dy vertical dimension.
00089         int ny = num_cells_y + 2; // Dy horizontal dimension.
00090         int mz = num_cells_z + 1; // Dz vertical dimension.
00091         int nz = num_cells_z + 2; // Dz horizontal dimension.
00092
00093         mtk::Div1D div;
00094
00095         bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00096
00097 #ifdef MTK_PERFORM_PREVENTIONS
00098     if (!info) {
00099         std::cerr << "Mimetic div could not be built." << std::endl;
00100         return info;
00101     }
00102 #endif
00103
00104         auto west = grid.west_bndy();
00105         auto east = grid.east_bndy();
00106         auto south = grid.south_bndy();
00107         auto north = grid.east_bndy();
00108         auto bottom = grid.bottom_bndy();
00109         auto top = grid.top_bndy();
00110
00111         mtk::UniStgGrid1D grid_x(west, east, num_cells_x,
00112             mtk::FieldNature::VECTOR);
00113         mtk::UniStgGrid1D grid_y(south, north, num_cells_y,
00114             mtk::FieldNature::VECTOR);
00115         mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z,
00116             mtk::FieldNature::VECTOR);
00117
00117         mtk::DenseMatrix Dx(div.ReturnAsDenseMatrix(grid_x));
00118         mtk::DenseMatrix Dy(div.ReturnAsDenseMatrix(grid_y));
00119         mtk::DenseMatrix Dz(div.ReturnAsDenseMatrix(grid_z));
00120
00121         bool padded{true};
00122         bool transpose{false};
00123
00124
00125         mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00126         mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00127         mtk::DenseMatrix iz(num_cells_z, padded, transpose);
00128
00129
00130         mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(iz, iy));
00131         mtk::DenseMatrix dx(mtk::DenseMatrix::Kron(aux1, Dx));
00132
00133         mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(iz, Dy));
00134         mtk::DenseMatrix dy(mtk::DenseMatrix::Kron(aux2, ix));
00135
00136
00137         mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Dz, iy));
00138         mtk::DenseMatrix dz(mtk::DenseMatrix::Kron(aux3, ix));
00139
00140 #if MTK_VERBOSE_LEVEL > 2
00141         std::cout << "Dx: " << mx << " by " << nx << std::endl;
00142         std::cout << "Ix: " << num_cells_x << " by " << nx << std::endl;
00143         std::cout << "Dy: " << my << " by " << ny << std::endl;
00144         std::cout << "Iy: " << num_cells_y << " by " << ny << std::endl;
00145         std::cout << "Dz: " << mz << " by " << nz << std::endl;
00146         std::cout << "Iz: " << num_cells_z << " by " << nz << std::endl;
00147 #endif
00148
00149
00150

```

```

00151     int total_rows{nx*ny*nz};
00152     int total_cols{mx*num_cells_y*num_cells_z +
00153                     num_cells_x*my*num_cells_z +
00154                     num_cells_x*num_cells_y*mz};
00155
00156 #if MTK_VERBOSE_LEVEL > 2
00157 std::cout << "Div 3D: " << total_rows << " by " << total_cols << std::endl;
00158#endif
00159
00160 mtk::DenseMatrix d3d(total_rows, total_cols);
00161
00162 for (auto ii = 0; ii < total_rows; ++ii) {
00163
00164     for (auto jj = 0; jj < mx*num_cells_y*num_cells_z; ++jj) {
00165         d3d.SetValue(ii, jj, dx.GetValue(ii, jj));
00166     }
00167
00168     int offset = mx*num_cells_y*num_cells_z;
00169
00170     for (auto kk = 0; kk < num_cells_x*my*num_cells_z; ++kk) {
00171         d3d.SetValue(ii, kk + offset, dy.GetValue(ii, kk));
00172     }
00173
00174     offset += num_cells_x*my*num_cells_z;
00175
00176     for (auto ll = 0; ll < num_cells_x*num_cells_y*mz; ++ll) {
00177         d3d.SetValue(ii, ll + offset, dz.GetValue(ii, ll));
00178     }
00179 }
00180
00181 divergence_ = d3d;
00182
00183 return info;
00184}
00185
00186 mtk::DenseMatrix mtk::Div3D::ReturnAsDenseMatrix() const {
00187
00188     return divergence_;
00189 }

```

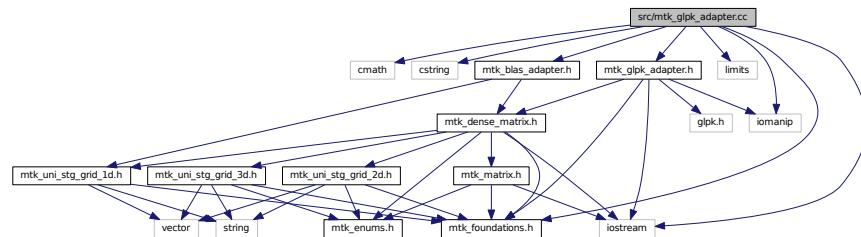
18.101 src/mtk_glpk_adapter.cc File Reference

Definition of an adapter class for the GLPK API.

```

#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include "mtk_foundations.h"
#include "mtk_blas_adapter.h"
#include "mtk_glpk_adapter.h"
Include dependency graph for mtk_glpk_adapter.cc:

```



18.101.1 Detailed Description

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

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

See also

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

Author

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

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

18.102 mtk_glpk_adapter.cc

```

00001
00020 /*
00021 Copyright (C) 2016, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065

```

```

00066 #include <cmath>
00067 #include <cstring>
00068
00069 #include <iostream>
00070 #include <iomanip>
00071 #include <limits>
00072
00073 #include "mtk_foundations.h"
00074 #include "mtk blas_adapter.h"
00075 #include "mtk_glpk_adapter.h"
00076
00077 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
    mtk::Real *A,
00078                                     int nrows,
00079                                     int ncols,
00080                                     int kk,
00081                                     mtk::Real *hh,
00082                                     mtk::Real *qq,
00083                                     int robjective,
00084                                     mtk::Real mimetic_threshold,
00085                                     int copy) noexcept {
00086
00087 #if MTK_DEBUG_LEVEL > 0
00088 char mps_file_name[18]; // File name for the MPS files.
00089 #endif
00090 char rname[5];           // Row name.
00091 char cname[5];           // Column name.
00092
00093 glp_prob *lp; // Linear programming problem.
00094
00095 int *ia;   // Array for the problem.
00096 int *ja;   // Array for the problem.
00097
00098 int problem_size; // Size of the problem.
00099 int lp_nrows;    // Number of rows.
00100 int lp_ncols;    // Number of columns.
00101 int matsize;     // Size of the matrix.
00102 int glp_index{1}; // Index of the objective function.
00103 int ii;          // Iterator.
00104 int jj;          // Iterator.
00105
00106 mtk::Real *ar;      // Array for the problem.
00107 mtk::Real *objective; // Array containing the objective function.
00108 mtk::Real *rhs;      // Array containing the rhs.
00109 mtk::Real *err;      // Array of errors.
00110
00111 mtk::Real x1;        // Norm-2 of the error.
00112
00113 #if MTK_DEBUG_LEVEL > 0
00114 mtk::Real obj_value; // Value of the objective function.
00115 #endif
00116
00117 lp_nrows = kk;
00118 lp_ncols = kk;
00119
00120 matsize = lp_nrows*lp_ncols;
00121
00122
00123
00124
00125 problem_size = lp_nrows*lp_ncols + 1;
00126
00127 try {
00128     ia = new int[problem_size];
00129 } catch (std::bad_alloc &memory_allocation_exception) {
00130     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00131     std::endl;
00132     std::cerr << memory_allocation_exception.what() << std::endl;
00133 }
00134
00135 memset(ia, 0, sizeof(ia[0])*problem_size);
00136
00137 try {
00138     ja = new int[problem_size];
00139 } catch (std::bad_alloc &memory_allocation_exception) {
00140     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00141     std::endl;
00142     std::cerr << memory_allocation_exception.what() << std::endl;
00143 }
00144
00145 memset(ja, 0, sizeof(ja[0])*problem_size);
00146
00147 try {
00148     ar = new mtk::Real[problem_size];

```

```

00148 } catch (std::bad_alloc &memory_allocation_exception) {
00149     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00150         std::endl;
00151     std::cerr << memory_allocation_exception.what() << std::endl;
00152 }
00153 memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00154
00155 try {
00156     objective = new mtk::Real[lp_ncols + 1];
00157 } catch (std::bad_alloc &memory_allocation_exception) {
00158     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00159         std::endl;
00160     std::cerr << memory_allocation_exception.what() << std::endl;
00161 }
00162 memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00163
00164 try {
00165     rhs = new mtk::Real[lp_nrows + 1];
00166 } catch (std::bad_alloc &memory_allocation_exception) {
00167     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00168         std::endl;
00169     std::cerr << memory_allocation_exception.what() << std::endl;
00170 }
00171 memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00172
00173 try {
00174     err = new mtk::Real[lp_nrows];
00175 } catch (std::bad_alloc &memory_allocation_exception) {
00176     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00177         std::endl;
00178     std::cerr << memory_allocation_exception.what() << std::endl;
00179 }
00180 memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00181
#if MTK_DEBUG_LEVEL > 0
00182 std::cout << "Problem size: " << problem_size << std::endl;
00183 std::cout << "lp_nrows = " << lp_nrows << std::endl;
00184 std::cout << "lp_ncols = " << lp_ncols << std::endl;
00185 std::cout << std::endl;
#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 glp_add_rows(lp, lp_nrows);
00194
00195 for (ii = 1; ii <= lp_nrows; ++ii) {
00196     sprintf(rname, "R%02d",ii);
00197     glp_set_row_name(lp, ii, rname);
00198 }
00199
00200 glp_add_cols(lp, lp_ncols);
00201
00202 for (ii = 1; ii <= lp_ncols; ++ii) {
00203     sprintf(cname, "Q%02d",ii);
00204     glp_set_col_name (lp, ii, cname);
00205 }
00206
#if MTK_DEBUG_LEVEL>0
00207 std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
#endif
00208
00209 for (jj = 0; jj < kk; ++jj) {
00210     objective[glp_index] = A[jj + robjective * ncols];
00211     glp_index++;
00212 }
00213
#if MTK_DEBUG_LEVEL >0
00214 std::cout << std::endl;
#endif
00215
00216 if (ii != robjective) {
00217     rhs[glp_index] = hh[ii];
00218     glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00219 }
00220
#if MTK_DEBUG_LEVEL >0
00221 std::cout << std::endl;
#endif
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     }
00231 }
```

```

00232     glp_index++;
00233 }
00234 }
00235
00236 #if MTK_DEBUG_LEVEL > 0
00237 std::cout << "rhs =" << std::endl;
00238 for (auto ii = 0; ii < lp_nrows; ++ii) {
00239   std::cout << std::setw(15) << rhs[ii] << std::endl;
00240 }
00241 std::cout << std::endl;
00242 #endif
00243
00244
00245 for (ii = 1; ii <= lp_ncols; ++ii) {
00246   glp_set_obj_coef (lp, ii, objective[ii]);
00247 }
00248
00249
00250
00251 for (ii = 1; ii <= lp_ncols; ++ii) {
00252   glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00253 }
00254
00255
00256
00257 glp_index = 1;
00258 for (ii = 0; ii <= kk; ++ii) {
00259   for (jj = 0; jj < kk; ++jj) {
00260     if (ii != robjective) {
00261       ar[glp_index] = A[jj + ii * ncols];
00262       glp_index++;
00263     }
00264   }
00265 }
00266 }
00267
00268 glp_index = 0;
00269
00270 for (ii = 1; ii < problem_size; ++ii) {
00271   if (((ii - 1) % lp_ncols) == 0) {
00272     glp_index++;
00273   }
00274   ia[iii] = glp_index;
00275   ja[iii] = (ii - 1) % lp_ncols + 1;
00276 }
00277
00278 glp_load_matrix(lp, matsize, ia, ja, ar);
00279
00280 #if MTK_DEBUG_LEVEL > 0
00281 sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00282 glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00283#endif
00284
00285
00286
00287 glp_simplex (lp, nullptr);
00288
00289 // Check status of the solution, determining if this was a feasible solution.
00290
00291 if (glp_get_status(lp) == GLP_OPT) {
00292
00293   for(ii = 1; ii <= lp_ncols; ++ii) {
00294     err[ii - 1] = qq[ii - 1] - glp_get_col_prim(lp,ii);
00295   }
00296
00297 #if MTK_DEBUG_LEVEL > 0
00298   obj_value = glp_get_obj_val (lp);
00299   std::cout << std::setw(12) << "CBS" << std::setw(12) << "CRS" << std::endl;
00300   for (ii = 0; ii < lp_ncols; ++ii) {
00301     std::cout << "q_" << ii + 1 << " = " << std::setw(12) <<
00302       glp_get_col_prim(lp,ii + 1) << std::setw(12) << qq[ii] << std::endl;
00303   }
00304   std::cout << "Objective function value (row " << robjective + 1 << ") = " <<
00305     obj_value << std::endl;
00306 #endif
00307
00308   if (copy) {
00309     for(ii = 0; ii < lp_ncols; ++ii) {
00310       qq[ii] = glp_get_col_prim(lp,ii + 1);
00311     }
00312     // Preserve the bottom values of qq.
00313   }
00314
00315   xl = mtk::BLASAdapter::RealNRM2(err,lp_ncols);
00316

```

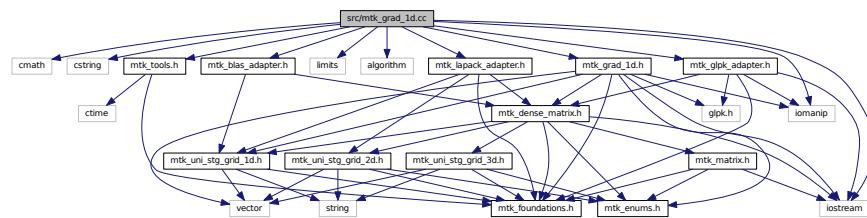
```

00317     } else {
00318         x1 = std::numeric_limits<mtk::Real>::infinity();
00319     }
00320
00321     glp_delete_prob (lp);
00322     glp_free_env ();
00323
00324     delete [] ia;
00325     delete [] ja;
00326     delete [] ar;
00327     delete [] objective;
00328     delete [] rhs;
00329     delete [] err;
00330
00331     return x1;
00332 }
```

18.103 src/mtk_grad_1d.cc File Reference

Implements the class Grad1D.

```
#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk blas adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_grad_1d.h"
Include dependency graph for mtk_grad_1d.cc:
```



Namespaces

- `mtk`

Mimetic Methods Toolkit namespace.

Functions

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

18.103.1 Detailed Description

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

Author

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

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

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

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

18.104 mtk_grad_1d.cc

```

00001
00015 /*
00016 Copyright (C) 2016, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066

```

```

00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075 #include "mtk blas_adapter.h"
00076 #include "mtk lapack_adapter.h"
00077 #include "mtk glpk_adapter.h"
00078 #include "mtk grad_ld.h"
00079
00080 namespace mtk {
00081
00082 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &in) {
00083
00084     int output_precision{4};
00085     int output_width{8};
00086
00088
00089     stream << "Order of accuracy: " << in.gradient_[0] << std::endl;
00090
00092
00093     stream << "Interior stencil: " << std::endl;
00094     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00095         stream << std::setprecision(output_precision) <<
00096             std::setw(output_width) << in.gradient_[ii] << ' ';
00097     }
00098     stream << std::endl;
00099
00101
00102     stream << "Weights:" << std::endl;
00103     for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00104         order_accuracy_; ++ii) {
00104         stream << std::setprecision(output_precision) <<
00105             std::setw(output_width) << in.gradient_[ii] << ' ';
00106     }
00107     stream << std::endl;
00108
00109
00110     int offset{2*in.order_accuracy_ + 1};
00111     int mm {};
00112
00113     if (in.order_accuracy_ > mtk::kDefaultOrderAccuracy) {
00114         for (auto ii = 0; ii < in.num_bndy_approx_ ; ++ii) {
00115             stream << "Boundary row " << ii + 1 << ":" << std::endl;
00116             for (auto jj = 0; jj < in.num_bndy_coeffs_ ; jj++) {
00117                 auto value = in.gradient_[offset + (mm)];
00118                 stream << std::setprecision(output_precision) <<
00119                     std::setw(output_width) << value << ' ';
00120                 mm++;
00121             }
00122             stream << std::endl;
00123             stream << "Sum of elements in boundary row " << ii + 1 << ":" <<
00124                 in.sums_rows_mim_bndy_[ii];
00125             stream << std::endl;
00126         }
00127     } else {
00128         stream << "Boundary row 1:" << std::endl;
00129         stream << std::setprecision(output_precision) <<
00130             std::setw(output_width) << in.gradient_[offset + 0] << ' ';
00131         stream << std::setprecision(output_precision) <<
00132             std::setw(output_width) << in.gradient_[offset + 1] << ' ';
00133         stream << std::setprecision(output_precision) <<
00134             std::setw(output_width) << in.gradient_[offset + 2] << ' ';
00135         stream << std::endl;
00136         stream << "Sum of elements in boundary row 1: " <<
00137             in.gradient_[offset + 0] + in.gradient_[offset + 1] +
00138             in.gradient_[offset + 2];
00139         stream << std::endl;
00140     }
00141
00142     return stream;
00143 }
00144 }
00145
00146 mtk::Grad1D::Grad1D():
00147     order_accuracy_(mtk::kDefaultOrderAccuracy),
00148     dim_null_(),
00149     num_bndy_approx_(),
00150     num_bndy_coeffs_()

```

```

00151     gradient_length_(),
00152     minrow_(),
00153     row_(),
00154     num_feasible_sols_(),
00155     coeffs_interior_(),
00156     prem_apps_(),
00157     weights_crs_(),
00158     weights_cbs_(),
00159     mim_bndy_(),
00160     gradient_(),
00161     mimetic_threshold_(mtk::kDefaultMimeticThreshold),
00162     mimetic_measure_(mtk::kZero),
00163     sums_rows_mim_bndy_() {}
00164
00165 mtk::Grad1D::Grad1D(const Grad1D &grad):
00166     order_accuracy_(grad.order_accuracy_),
00167     dim_null_(grad.dim_null_),
00168     num_bndy_approxos_(grad.num_bndy_approxos_),
00169     num_bndy_coeffs_(grad.num_bndy_coeffs_),
00170     gradient_length_(grad.gradient_length_),
00171     minrow_(grad.minrow_),
00172     row_(grad.row_),
00173     num_feasible_sols_(grad.num_feasible_sols_),
00174     coeffs_interior_(grad.coeffs_interior_),
00175     prem_apps_(grad.prem_apps_),
00176     weights_crs_(grad.weights_crs_),
00177     weights_cbs_(grad.weights_cbs_),
00178     mim_bndy_(grad.mim_bndy_),
00179     gradient_(grad.gradient_),
00180     mimetic_threshold_(grad.mimetic_threshold_),
00181     mimetic_measure_(grad.mimetic_measure_),
00182     sums_rows_mim_bndy_(grad.sums_rows_mim_bndy_) {}
00183
00184 mtk::Grad1D::~Grad1D() {
00185
00186     delete[] coeffs_interior_;
00187     coeffs_interior_ = nullptr;
00188
00189     delete[] prem_apps_;
00190     prem_apps_ = nullptr;
00191
00192     delete[] weights_crs_;
00193     weights_crs_ = nullptr;
00194
00195     delete[] weights_cbs_;
00196     weights_cbs_ = nullptr;
00197
00198     delete[] mim_bndy_;
00199     mim_bndy_ = nullptr;
00200
00201     delete[] gradient_;
00202     gradient_ = nullptr;
00203 }
00204
00205 bool mtk::Grad1D::ConstructGrad1D(int order_accuracy,
00206                                     Real mimetic_threshold) {
00207
00208 #ifdef MTK_PERFORM_PREVENTIONS
00209     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00210     mtk::Tools::Prevent((order_accuracy*2) != 0, __FILE__, __LINE__, __func__);
00211     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00212                         __FILE__, __LINE__, __func__);
00213
00214     if (order_accuracy >= mtk::kCriticalOrderAccuracyGrad) {
00215         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00216     }
00217
00218     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00219     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00220     #endif
00221
00222     order_accuracy_ = order_accuracy;
00223     mimetic_threshold_ = mimetic_threshold;
00224
00225     bool abort_construction = ComputeStencilInteriorGrid();
00226
00227 #ifdef MTK_PERFORM_PREVENTIONS
00228     if (!abort_construction) {
00229         std::cerr << "Could NOT complete stage 1." << std::endl;
00230         std::cerr << "Exiting..." << std::endl;
00231         return false;
00231 
```

```

00232     }
00233 #endif
00234
00235 // At this point, we already have the values for the interior stencil stored
00236 // in the coeffs_interior_ array.
00237
00238 dim_null_ = order_accuracy_/2 - 1;
00239
00240 num_bndy_approxs_ = dim_null_ + 1;
00241
00242 #ifdef MTK_PRECISION_DOUBLE
00243 num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00244 #else
00245 num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00246 #endif
00247
00248
00249 // For this we will follow recommendations given in:
00250 //
00251 // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00252 //
00253 // We will compute the QR Factorization of the transpose, as in the
00254 // following (MATLAB) pseudo-code:
00255 //
00256 //
00257 // [Q,R] = qr(V'); % Full QR as defined in
00258 // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00259 //
00260 // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00261 //
00262 // However, given the nature of the Vandermonde matrices we've just
00263 // computed, they all posses the same null-space. Therefore, we impose the
00264 // convention of computing the null-space of the first Vandermonde matrix
00265 // (west boundary).
00266
00267 // In the case of the gradient, the first Vandermonde system has a unique
00268 // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00269 // matrix used to assemble said system, will have an empty null-space.
00270
00271 // Therefore, we only compute a rational basis for the case of order higher
00272 // than second.
00273
00274 if (dim_null_ > 0) {
00275
00276     abort_construction = ComputeRationalBasisNullSpace();
00277
00278 #ifdef MTK_PERFORM_PREVENTIONS
00279     if (!abort_construction) {
00280         std::cerr << "Could NOT complete stage 2.1." << std::endl;
00281         std::cerr << "Exiting..." << std::endl;
00282         return false;
00283     }
00284 #endif
00285 }
00286
00287 abort_construction = ComputePreliminaryApproximations();
00288
00289 #ifdef MTK_PERFORM_PREVENTIONS
00290     if (!abort_construction) {
00291         std::cerr << "Could NOT complete stage 2.2." << std::endl;
00292         std::cerr << "Exiting..." << std::endl;
00293         return false;
00294     }
00295 #endif
00296
00297 abort_construction = ComputeWeights();
00298
00299 #ifdef MTK_PERFORM_PREVENTIONS
00300     if (!abort_construction) {
00301         std::cerr << "Could NOT complete stage 2.3." << std::endl;
00302         std::cerr << "Exiting..." << std::endl;
00303         return false;
00304     }
00305 #endif
00306
00307 #endif
00308
00309 if (dim_null_ > 0) {
00310
00311     abort_construction = ComputeStencilBoundaryGrid();
00312
00313 #ifdef MTK_PERFORM_PREVENTIONS
00314     if (!abort_construction) {
00315         std::cerr << "Could NOT complete stage 2.4." << std::endl;
00316

```

```

00317     std::cerr << "Exiting..." << std::endl;
00318     return false;
00319 }
00320 #endif
00321 }
00322
00323
00324 // Once we have the following three collections of data:
00325 // (a) the coefficients for the interior,
00326 // (b) the coefficients for the boundary (if it applies),
00327 // (c) and the weights (if it applies),
00328 // we will store everything in the output array:
00329
00330 abort_construction = AssembleOperator();
00331
00332
00333 #ifdef MTK_PERFORM_PREVENTIONS
00334 if (!abort_construction) {
00335     std::cerr << "Could NOT complete stage 3." << std::endl;
00336     std::cerr << "Exiting..." << std::endl;
00337     return false;
00338 }
00339 #endif
00340
00341 return true;
00342 }
00343
00344 int mtk::Grad1D::num_bndy_coeffs() const {
00345     return num_bndy_coeffs_;
00346 }
00347
00348
00349 mtk::Real *mtk::Grad1D::coeffs_interior() const {
00350     return coeffs_interior_;
00351 }
00352
00353
00354 mtk::Real *mtk::Grad1D::weights_crs() const {
00355     return weights_crs_;
00356 }
00357
00358
00359 mtk::Real *mtk::Grad1D::weights_cbs() const {
00360     return weights_cbs_;
00361 }
00362
00363
00364 int mtk::Grad1D::num_feasible_sols() const {
00365     return num_feasible_sols_;
00366 }
00367
00368
00369 mtk::DenseMatrix mtk::Grad1D::mim_bndy() const {
00370
00371     mtk::DenseMatrix xx(dim_null_ + 1, 3*order_accuracy_/2);
00372
00373     auto counter = 0;
00374     for (auto ii = 0; ii < dim_null_ + 1; ++ii) {
00375         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00376             xx.SetValue(ii,jj, gradient_[2*order_accuracy_ + 1 + counter]);
00377             counter++;
00378         }
00379     }
00380
00381     return xx;
00382 }
00383
00384 std::vector<mtk::Real> mtk::Grad1D::sums_rows_mim_bndy() const {
00385
00386     return sums_rows_mim_bndy_;
00387 }
00388
00389 mtk::Real mtk::Grad1D::mimetic_measure() const {
00390
00391     return mimetic_measure_;
00392 }
00393
00394 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00395     mtk::Real east,
00396     int num_cells_x) const {
00397

```

```

00398 int nn{num_cells_x}; // Number of cells on the grid.
00399
00400 #ifdef MTK_PERFORM_PREVENTIONS
00401 mtk::Tools::Prevent(east < west, __FILE__, __LINE__, __func__);
00402 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00403 mtk::Tools::Prevent(nn < 3*order_accuracy_- 2, __FILE__, __LINE__, __func__);
00404 #endif
00405
00406 mtk::Real delta_x = (east - west)/((mtk::Real) num_cells_x);
00407
00408 mtk::Real inv_delta_x{mtk::kOne/delta_x};
00409
00410 int gg_num_rows = nn + 1;
00411 int gg_num_cols = nn + 2;
00412 int num_extra_rows = order_accuracy_-2;
00413 int elements_per_extra_row = num_bndy_coefffs_;
00414
00415 // Output matrix featuring sizes for gradient operators.
00416 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00417
00418 out.set_encoded_operator(mtk::EncodedOperator::GRADIENT
    );
00419
00420
00421 auto ee_index = 0;
00422 for (auto ii = 0; ii < num_extra_rows; ii++) {
00423     auto cc = 0;
00424     for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00425         if(cc >= elements_per_extra_row) {
00426             out.SetValue(ii, jj, mtk::kZero);
00427         } else {
00428             out.SetValue(ii,jj,
00429                         gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00430             cc++;
00431         }
00432     }
00433 }
00434
00435
00436
00437 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00438     auto jj = ii - num_extra_rows + 1;
00439     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00440         out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00441     }
00442 }
00443
00444
00445 ee_index = 0;
00446 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00447     auto cc = 0;
00448     for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00449         if(cc >= elements_per_extra_row) {
00450             out.SetValue(ii,jj,mtk::kZero);
00451         } else {
00452             out.SetValue(ii,jj,
00453                         -gradient_[2*order_accuracy_ + 1 +
00454 ee_index++]*inv_delta_x);
00455             cc++;
00456         }
00457     }
00458 }
00459
00460 }
00461
00462 return out;
00463 }
00464
00465 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00466     const UniStgGrid1D &grid) const {
00467
00468     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00469
00470 #ifdef MTK_PERFORM_PREVENTIONS
00471 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00472 mtk::Tools::Prevent(nn < 3*order_accuracy_- 2, __FILE__, __LINE__, __func__);
00473 mtk::Tools::Prevent(grid.field_nature() !=
00474                     mtk::FieldNature::SCALAR,
00475                     __FILE__, __LINE__, __func__);
00476 #endif
00477
00478     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00479
00480     int gg_num_rows = nn + 1;

```

```

00480 int gg_num_cols = nn + 2;
00481 int num_extra_rows = order_accuracy_/2;
00482 int elements_per_row = num_bndy_coeffs_;
00483
00484 // Output matrix featuring sizes for gradient operators.
00485 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00486
00487 out.set_encoded_operator(mtk::EncodedOperator::GRADIENT
);
00488
00489
00490 auto ee_index = 0;
00491 for (auto ii = 0; ii < num_extra_rows; ii++) {
00492     auto cc = 0;
00493     for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00494         if(cc >= elements_per_row) {
00495             out.SetValue(ii, jj, mtk::kZero);
00496         } else {
00497             out.SetValue(ii,jj,
00498                         gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00499             cc++;
00500         }
00501     }
00502 }
00503
00504
00505 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00506     auto jj = ii - num_extra_rows + 1;
00507     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00508         out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00509     }
00510 }
00511
00512
00513
00514 ee_index = 0;
00515 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00516     auto cc = 0;
00517     for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00518         if(cc >= elements_per_row) {
00519             out.SetValue(ii,jj,mtk::kZero);
00520         } else {
00521             out.SetValue(ii,jj,
00522                         -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00523             cc++;
00524         }
00525     }
00526 }
00527
00528
00529 return out;
00530
00531 }
00532
00533 mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix
(
00534     int num_cells_x) const {
00535
00536     int nn{num_cells_x}; // Number of cells on the grid.
00537
00538 #ifdef MTK_PERFORM_PREVENTIONS
00539 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00540 mtk::Tools::Prevent(nn < 3*order_accuracy_- 2, __FILE__, __LINE__, __func__);
00541 #endif
00542
00543     int gg_num_rows = nn + 1;
00544     int gg_num_cols = nn + 2;
00545     int elements_per_extra_row = num_bndy_coeffs_;
00546     int num_extra_rows = order_accuracy_/2;
00547
00548 // Output matrix featuring sizes for gradient operators.
00549 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00550
00551 out.set_encoded_operator(mtk::EncodedOperator::GRADIENT
);
00552
00553
00554 auto ee_index = 0;
00555 for (auto ii = 0; ii < num_extra_rows; ii++) {
00556     auto cc = 0;
00557     for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00558         if(cc >= elements_per_extra_row) {
00559             out.SetValue(ii, jj, mtk::kZero);
00560         } else {
00561             out.SetValue(ii,jj,
00562                         gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00563             cc++;
00564         }
00565     }
00566 }
00567
00568
00569 return out;

```

```

00562     out.SetValue(ii, jj,
00563                 gradient_[2*order_accuracy_ + 1 + ee_index++]);
00564     cc++;
00565   }
00566 }
00568
00569
00570 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00571   auto jj = ii - num_extra_rows + 1;
00572   for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00573     out.SetValue(ii, jj, coeffs_interior_[cc]);
00574   }
00575 }
00576
00577
00578 ee_index = 0;
00579 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00580   auto cc = 0;
00581   for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00582     if(cc >= elements_per_extra_row) {
00583       out.SetValue(ii,jj,mtk::kZero);
00584     } else {
00585       out.SetValue(ii,jj,
00586                   -gradient_[2*order_accuracy_ + 1 + ee_index++]);
00587     }
00588     cc++;
00589   }
00590 }
00591
00592 }
00593
00594 return out;
00595 }
00596
00597 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00598
00599
00600 mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00601
00602 try {
00603   pp = new mtk::Real[order_accuracy_];
00604 } catch (std::bad_alloc &memory_allocation_exception) {
00605   std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00606   std::endl;
00607   std::cerr << memory_allocation_exception.what() << std::endl;
00608 }
00609 memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00610
00611 #ifdef MTK_PRECISION_DOUBLE
00612 pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00613 #else
00614 pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00615 #endif
00616
00617 for (auto ii = 1; ii < order_accuracy_; ++ii) {
00618   pp[ii] = pp[ii - 1] + mtk::kOne;
00619 }
00620
00621
00622 #if MTK_VERBOSE_LEVEL > 3
00623 std::cout << "pp =" << std::endl;
00624 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00625   std::cout << std::setw(12) << pp[ii];
00626 }
00627 std::cout << std::endl << std::endl;
00628 #endif
00629
00630
00631 bool transpose{false};
00632
00633 mtk::DenseMatrix vander_matrix(pp,order_accuracy_,order_accuracy_,transpose);
00634
00635 #if MTK_VERBOSE_LEVEL > 4
00636 std::cout << "vander_matrix = " << std::endl;
00637 std::cout << vander_matrix << std::endl << std::endl;
00638 #endif
00639
00640
00641
00642
00643 try {
00644   coeffs_interior_ = new mtk::Real[order_accuracy_];
00645 } catch (std::bad_alloc &memory_allocation_exception) {
00646   std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00647   std::endl;

```

```

00648     std::cerr << memory_allocation_exception.what() << std::endl;
00649 }
00650 memset(coeffs_interior_, mtk::kZero,
00651 sizeof(coeffs_interior_[0])*order_accuracy_);
00652
00653 coeffs_interior_[1] = mtk::kOne;
00654
00655 #if MTK_VERBOSE_LEVEL > 3
00656 std::cout << "oo =" << std::endl;
00657 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00658     std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00659 }
00660 std::cout << std::endl;
00661 #endif
00662
00663
00664 int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00665                                     coeffs_interior_)};
00666
00667 #ifdef MTK_PERFORM_PREVENTIONS
00668 if (!info) {
00669     std::cout << "System solved! Interior stencil attained!" << std::endl;
00670     std::cout << std::endl;
00671 }
00672 }
00673 else {
00674     std::cerr << "Something wrong solving system! info = " << info << std::endl;
00675     std::cerr << "Exiting..." << std::endl;
00676     return false;
00677 }
00678 #endif
00679
00680 #if MTK_VERBOSE_LEVEL > 3
00681 std::cout << "coeffs_interior_ =" << std::endl;
00682 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00683     std::cout << std::setw(12) << coeffs_interior_[ii];
00684 }
00685 std::cout << std::endl << std::endl;
00686 #endif
00687
00688 delete [] pp;
00689 pp = nullptr;
00690
00691 return true;
00692 }
00693
00694 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00695
00696
00697 mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00698
00699 try {
00700     gg = new mtk::Real[num_bndy_coeffs_];
00701 } catch (std::bad_alloc &memory_allocation_exception) {
00702     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00703     std::endl;
00704     std::cerr << memory_allocation_exception.what() << std::endl;
00705 }
00706
00707 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00708
00709 #ifdef MTK_PRECISION_DOUBLE
00710 gg[1] = 1.0/2.0;
00711 #else
00712 gg[1] = 1.0f/2.0f;
00713 #endif
00714 for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00715     gg[ii] = gg[ii - 1] + mtk::kOne;
00716 }
00717
00718 #if MTK_VERBOSE_LEVEL > 3
00719 std::cout << "gg =" << std::endl;
00720 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00721     std::cout << std::setw(12) << gg[ii];
00722 }
00723 std::cout << std::endl << std::endl;
00724 #endif
00725
00726
00727 bool tran{true}; // Should I transpose the Vandermonde matrix.
00728
00729 mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00730
00731

```

```

00732 #if MTK_VERBOSE_LEVEL > 4
00733 std::cout << "aa_west_t =" << std::endl;
00734 std::cout << aa_west_t << std::endl;
00735 #endif
00736
00738
00739 mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00740 (aa_west_t));
00741 #if MTK_VERBOSE_LEVEL > 3
00742 std::cout << "qq_t = " << std::endl;
00743 std::cout << qq_t << std::endl;
00744 #endif
00745
00747
00748 int kk_num_rows{num_bndy_coeffs_};
00749 int kk_num_cols{dim_null_};
00750
00751 mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00752
00753 // In the case of the gradient, even though we must solve for a null-space
00754 // of dimension 2, we must only extract ONE basis for the kernel.
00755 // We perform this extraction here:
00756
00757 int aux_{kk_num_rows - kk_num_cols};
00758 for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00759     aux--;
00760     for (auto jj = 0; jj < kk_num_rows; jj++) {
00761         kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux - 1)] =
00762             qq_t.data()[ii*num_bndy_coeffs_ + jj];
00763     }
00764 }
00765
00766 #if MTK_VERBOSE_LEVEL > 2
00767 std::cout << "kk =" << std::endl;
00768 std::cout << kk << std::endl;
00769 std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00770 std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00771 std::cout << std::endl;
00772 #endif
00773
00774
00775 // Scale thus requesting that the last entries of the attained basis for the
00776 // null-space, adopt the pattern we require.
00777 // Essentially we will implement the following MATLAB pseudo-code:
00778 // scalers = kk(num_bndy_approx - (dim_null - 1):num_bndy_approx,:)\B
00779 // SK = kk*scalers
00780 // where SK is the scaled null-space.
00781
00782 // In this point, we almost have all the data we need correctly allocated
00783 // in memory. We will create the matrix iden_, and elements we wish to scale
00784 // in the kk array. Using the concept of the leading dimension, we could just
00785 // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00786 // GET how does it work. So I will just create a matrix with the content of
00787 // this array that we need, solve for the scalers and then scale the
00788 // whole kk:
00789
00790 // We will then create memory for that sub-matrix of kk (subk).
00791
00792
00793 mtk::DenseMatrix subk(dim_null_, dim_null_);
00794
00795 auto zz = 0;
00796 for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00797     for (auto jj = 0; jj < dim_null_; jj++) {
00798         subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00799     }
00800     zz++;
00801 }
00802
00803 #if MTK_VERBOSE_LEVEL > 4
00804 std::cout << "subk =" << std::endl;
00805 std::cout << subk << std::endl;
00806 #endif
00807
00808 subk.Transpose();
00809
00810 #if MTK_VERBOSE_LEVEL > 4
00811 std::cout << "subk_t =" << std::endl;
00812 std::cout << subk << std::endl;
00813 #endif
00814

```

```

00815     bool padded{false};
00816     tran = false;
00817
00818     mtk::DenseMatrix iden(dim_null_, padded, tran);
00819
00820 #if MTK_VERBOSE_LEVEL > 4
00821     std::cout << "iden =" << std::endl;
00822     std::cout << iden << std::endl;
00823 #endif
00824
00825 // Solve the system to compute the scalers.
00826 // An example of the system to solve, for k = 8, is:
00827 //
00828 // subk*scalers = iden or
00829 //
00830 // | 0.386018 -0.0339244 -0.129478 |      | 1 0 0 |
00831 // | -0.119774  0.0199423  0.0558632 |*scalers = | 0 1 0 |
00832 // | 0.0155708 -0.00349546 -0.00853182 |      | 0 0 1 |
00833 //
00834 // Notice this is a nrhs = 3 system.
00835 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00836 // will be stored in the created identity matrix.
00837 // Let us first transpose subk (because of LAPACK):
00838
00839     int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00840
00841 #ifdef MTK_PERFORM_PREVENTIONS
00842     if (!info) {
00843         std::cout << "System successfully solved!" <<
00844             std::endl;
00845     } else {
00846         std::cerr << "Something went wrong solving system! info = " << info <<
00847             std::endl;
00848         std::cerr << "Exiting..." << std::endl;
00849         return false;
00850     }
00851     std::cout << std::endl;
00852 #endif
00853
00854 #if MTK_VERBOSE_LEVEL > 4
00855     std::cout << "Computed scalers:" << std::endl;
00856     std::cout << iden << std::endl;
00857 #endif
00858
00859 // Multiply the two matrices to attain a scaled basis for null-space.
00860
00861     rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00862
00863 #if MTK_VERBOSE_LEVEL > 4
00864     std::cout << "Rational basis for the null-space:" << std::endl;
00865     std::cout << rat_basis_null_space_ << std::endl;
00866 #endif
00867
00868 // At this point, we have a rational basis for the null-space, with the
00869 // pattern we need! :)
00870
00871     delete [] gg;
00872     gg = nullptr;
00873
00874     return true;
00875 }
00876
00877 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00878
00879     mtk::Real *gg{}; // Generator vector for the first approximation.
00880
00881     try {
00882         gg = new mtk::Real[num_bndy_coeffs_];
00883     } catch (std::bad_alloc &memory_allocation_exception) {
00884         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00885             std::endl;
00886         std::cerr << memory_allocation_exception.what() << std::endl;
00887     }
00888     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00889
00890 #ifdef MTK_PRECISION_DOUBLE
00891     gg[1] = 1.0/2.0;
00892 #else
00893     gg[1] = 1.0f/2.0f;
00894 #endif

```

```

00897     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00898         gg[ii] = gg[ii - 1] + mtk::kOne;
00899     }
00900
00901 #if MTK_VERBOSE_LEVEL > 3
00902 std::cout << "gg0 =" << std::endl;
00903 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00904     std::cout << std::setw(12) << gg[ii];
00905 }
00906 std::cout << std::endl << std::endl;
00907 #endif
00908
00909 // Allocate 2D array to store the collection of preliminary approximations.
00910 try {
00911     prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00912 } catch (std::bad_alloc &memory_allocation_exception) {
00913     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00914 std::endl;
00915     std::cerr << memory_allocation_exception.what() << std::endl;
00916 }
00917 memset(prem_apps_,
00918         mtk::kZero,
00919         sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00920
00921
00922 for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00923
00924     // Re-check new generator vector for every iteration except for the first.
00925 #if MTK_VERBOSE_LEVEL > 3
00926 if (ll > 0) {
00927     std::cout << "gg_" << ll << " =" << std::endl;
00928     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00929         std::cout << std::setw(12) << gg[ii];
00930     }
00931     std::cout << std::endl << std::endl;
00932 }
00933 #endif
00934
00935
00936
00937 bool transpose{false};
00938
00939
00940 mtk::DenseMatrix aa(gg,
00941                     num_bndy_coeffs_, order_accuracy_ + 1,
00942                     transpose);
00943
00944 #if MTK_VERBOSE_LEVEL > 4
00945 std::cout << "aa_" << ll << " =" << std::endl;
00946 std::cout << aa << std::endl;
00947 #endif
00948
00949
00950
00951 mtk::Real *ob{};
00952
00953 auto ob_ld = num_bndy_coeffs_;
00954
00955 try {
00956     ob = new mtk::Real[ob_ld];
00957 } catch (std::bad_alloc &memory_allocation_exception) {
00958     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00959     std::endl;
00960     std::cerr << memory_allocation_exception.what() << std::endl;
00961 }
00962 memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00963
00964 ob[1] = mtk::kOne;
00965
00966 #if MTK_VERBOSE_LEVEL > 3
00967 std::cout << "ob = " << std::endl << std::endl;
00968 for (auto ii = 0; ii < ob_ld; ++ii) {
00969     std::cout << std::setw(12) << ob[ii] << std::endl;
00970 }
00971 std::cout << std::endl;
00972 #endif
00973
00974
00975
00976 // However, this is an under-determined system of equations. So we can not
00977 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00978 // our LAPACKAdapter class.
00979
00980 int info_{
00981     mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob

```

```

        , ob_ld);
00982
00983     #ifdef MTK_PERFORM_PREVENTIONS
00984     if (!info_) {
00985         std::cout << "System successfully solved!" << std::endl << std::endl;
00986     } else {
00987         std::cerr << "Error solving system! info = " << info_ << std::endl;
00988         return false;
00989     }
00990 #endif
00991
00992     #if MTK_VERBOSE_LEVEL > 3
00993     std::cout << "ob =" << std::endl;
00994     for (auto ii = 0; ii < ob_ld; ++ii) {
00995         std::cout << std::setw(12) << ob[ii] << std::endl;
00996     }
00997     std::cout << std::endl;
00998 #endif
00999
01000
01001 // This implies a DAXPY operation. However, we must construct the arguments
01002 // for this operation.
01003
01004 // Save them into the ob_bottom array:
01005
01006 Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
01007
01008 try {
01009     ob_bottom = new mtk::Real[dim_null_];
01010 } catch (std::bad_alloc &memory_allocation_exception) {
01011     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01012         std::endl;
01013     std::cerr << memory_allocation_exception.what() << std::endl;
01014 }
01015 memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
01016
01017 for (auto ii = 0; ii < dim_null_; ++ii) {
01018     ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
01019 }
01020
01021
01022 #if MTK_VERBOSE_LEVEL > 3
01023 std::cout << "ob_bottom =" << std::endl;
01024 for (auto ii = 0; ii < dim_null_; ++ii) {
01025     std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
01026 }
01027 std::cout << std::endl;
01028 #endif
01029
01030
01031
01032 // We must computed an scaled ob, sob, using the scaled null-space in
01033 // rat_basis_null_space_.
01034 // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
01035 // or:           ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
01036 // thus:          Y =   a*A    *x      +   b*Y (DAXPY).
01037
01038
01039 #if MTK_VERBOSE_LEVEL > 4
01040 std::cout << "Rational basis for the null-space:" << std::endl;
01041 std::cout << rat_basis_null_space_ << std::endl;
01042 #endif
01043
01044 mtk::Real alpha{-mtk::kOne};
01045 mtk::Real beta{mtk::kOne};
01046
01047 mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
01048                                     ob_bottom, beta, ob);
01049
01050 #if MTK_VERBOSE_LEVEL > 3
01051 std::cout << "scaled ob:" << std::endl;
01052 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01053     std::cout << std::setw(12) << ob[ii] << std::endl;
01054 }
01055 std::cout << std::endl;
01056 #endif
01057
01058 // We save the recently scaled solution, into an array containing these.
01059 // We can NOT start building the pi matrix, simply because I want that part
01060 // to be separated since its construction depends on the algorithm we want
01061 // to implement.
01062
01063 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01064     prem_apps_[ii*num_bndy_approx_ + 11] = ob[ii];
01065

```

```

01065     }
01066
01067     // After the first iteration, simply shift the entries of the last
01068     // generator vector used:
01069     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01070         gg[ii]--;
01071     }
01072
01073     // Garbage collection for this loop:
01074     delete[] ob;
01075     ob = nullptr;
01076
01077     delete[] ob_bottom;
01078     ob_bottom = nullptr;
01079 } // End of: for (ll = 0; ll < dim_null; ll++);

01080
01081 #if MTK_VERBOSE_LEVEL > 4
01082 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
01083 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01084     for (auto jj = 0; jj < num_bndy_approxos_; ++jj) {
01085         std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxos_ + jj];
01086     }
01087     std::cout << std::endl;
01088 }
01089 std::cout << std::endl;
01090 #endif
01091
01092 delete[] gg;
01093 gg = nullptr;
01094
01095 return true;
01096 }
01097
01098 bool mtk::Grad1D::ComputeWeights() {
01099
01100     // Matrix to compute the weights as in the CRSA.
01101     mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
01102
01103
01104     // Assemble the pi matrix using:
01105     // 1. The collection of scaled preliminary approximations.
01106     // 2. The collection of coefficients approximating at the interior.
01107     // 3. The scaled basis for the null-space.
01108
01109     // 1.1. Process array of scaled preliminary approximations.
01110
01111     // These are queued in scaled_solutions. Each one of these, will be a column
01112     // of the pi matrix:
01113     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01114         for (auto jj = 0; jj < num_bndy_approxos_; ++jj) {
01115             pi.data()[ii*(2*(num_bndy_approxos_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01116                 prem_apps_[ii*num_bndy_approxos_ + jj];
01117         }
01118     }
01119
01120     // 1.2. Add columns from known stencil approximating at the interior.
01121
01122     // However, these must be padded by zeros, according to their position in the
01123     // final pi matrix:
01124     auto mm = 1;
01125     for (auto jj = num_bndy_approxos_; jj < order_accuracy_; ++jj) {
01126         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01127             auto de = (ii + mm)*(2*(num_bndy_approxos_ - 1) +
01128                 (order_accuracy_/2 + 1)) + jj;
01129             pi.data()[de] = coeffs_interior_[ii];
01130         }
01131         ++mm;
01132     }
01133
01134     rat_basis_null_space_.OrderColMajor();
01135
01136     #if MTK_VERBOSE_LEVEL > 4
01137     std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01138     std::cout << rat_basis_null_space_ << std::endl;
01139     #endif
01140
01141     // 1.3. Add final set of columns: rational basis for null-space.
01142
01143     for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01144         jj < num_bndy_coeffs_ - 1; ++jj) {
01145         for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01146

```

```

01147     auto og =
01148         (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01149     auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01150     pi.data()[de] = rat_basis_null_space_.data()[og];
01151 }
01152 }
01153
01154 #if MTK_VERBOSE_LEVEL > 4
01155 std::cout << "coeffs_interior_ =" << std::endl;
01156 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01157     std::cout << std::setw(12) << coeffs_interior_[ii];
01158 }
01159 std::cout << std::endl << std::endl;
01160 #endif
01161
01162 #if MTK_VERBOSE_LEVEL > 4
01163 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01164 std::cout << pi << std::endl;
01165 #endif
01166
01167
01168 // This imposes the mimetic condition.
01169
01170 mtk::Real *hh{}; // Right-hand side to compute weights in the C(R,B)SA.
01171
01172 try {
01173     hh = new mtk::Real[num_bndy_coeffs_];
01174 } catch (std::bad_alloc &memory_allocation_exception) {
01175     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01176         std::endl;
01177     std::cerr << memory_allocation_exception.what() << std::endl;
01178 }
01179 memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01180
01181 hh[0] = -mtk::kOne;
01182 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01183     auto aux_xx = mtk::kZero;
01184     for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01185         aux_xx += coeffs_interior_[jj];
01186     }
01187     hh[ii] = -mtk::kOne*aux_xx;
01188 }
01189
01190
01191 // That is, we construct a system, to solve for the weights.
01192
01193 // Once again we face the challenge of solving with LAPACK. However, for the
01194 // CRSAs, this matrix PI is over-determined, since it has more rows than
01195 // unknowns. However, according to the theory, the solution to this system is
01196 // unique. We will use dgels_.
01197
01198 try {
01199     weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01200 } catch (std::bad_alloc &memory_allocation_exception) {
01201     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01202         std::endl;
01203     std::cerr << memory_allocation_exception.what() << std::endl;
01204 }
01205 memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01206
01207 int weights_ld{pi.num_cols() + 1};
01208
01209 // Preserve hh.
01210 std::copy(hh, hh + weights_ld, weights_cbs_);
01211
01212 pi.Transpose();
01213
01214 int info{
01215     mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01216                                         weights_cbs_, weights_ld)
01217 };
01218
01219 #ifdef MTK_PERFORM_PREVENTIONS
01220 if (!info) {
01221     std::cout << "System successfully solved!" << std::endl << std::endl;
01222 } else {
01223     std::cerr << "Error solving system! info = " << info << std::endl;
01224     return false;
01225 }
01226
01227 #endif
01228
01229

```

```

01230 #if MTK_VERBOSE_LEVEL > 3
01231 std::cout << "hh =" << std::endl;
01232 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01233     std::cout << std::setw(11) << hh[ii] << std::endl;
01234 }
01235 std::cout << std::endl;
01236 #endif
01237
01238 // Preserve the original weights for research.
01239
01240 try {
01241     weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01242 } catch (std::bad_alloc &memory_allocation_exception) {
01243     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01244     std::endl;
01245     std::cerr << memory_allocation_exception.what() << std::endl;
01246 }
01247 memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01248
01249 std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01250
01251 #if MTK_VERBOSE_LEVEL > 3
01252 std::cout << "weights_CRSA + lambda =" << std::endl;
01253 for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01254     std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01255 }
01256 std::cout << std::endl;
01257 #endif
01258
01259 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01260
01261     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01262
01263     // 6.1. Insert preliminary approximations to first set of columns.
01264
01265     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01266         for (auto jj = 0; jj < num_bndy_approxxs_; ++jj) {
01267             phi.data()[ii*(order_accuracy_) + jj] =
01268                 prem_apps_[ii*num_bndy_approxxs_ + jj];
01269         }
01270     }
01271
01272     // 6.2. Skip a column and negate preliminary approximations.
01273
01274     for (auto jj = 0; jj < order_accuracy_ + 1; jj++) {
01275         for (auto ii = 1; ii < num_bndy_approxxs_; ii++) {
01276             auto de = (ii*order_accuracy_ - num_bndy_approxxs_ + jj*order_accuracy_);
01277             auto og = (num_bndy_approxxs_ - ii + (jj)*num_bndy_approxxs_);
01278             phi.data()[de] = -prem_apps_[og];
01279         }
01280     }
01281
01282     // 6.3. Flip negative columns up-down.
01283
01284     for (auto ii = 0; ii < order_accuracy_/2; ii++) {
01285         for (auto jj = num_bndy_approxxs_ + 1; jj < order_accuracy_; jj++) {
01286             auto aux = phi.data()[ii*order_accuracy_ + jj];
01287             phi.data()[ii*order_accuracy_ + jj] =
01288                 phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01289             phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01290         }
01291     }
01292
01293     // 6.4. Insert stencil.
01294
01295     auto mm = 0;
01296     for (auto jj = num_bndy_approxxs_; jj < num_bndy_approxxs_ + 1; jj++) {
01297         for (auto ii = 0; ii < order_accuracy_ + 1; ii++) {
01298             if (ii == 0) {
01299                 phi.data()[jj] = 0.0;
01300             } else {
01301                 phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior_[ii - 1];
01302             }
01303         }
01304         mm++;
01305     }
01306
01307     #if MTK_VERBOSE_LEVEL > 4
01308     std::cout << "phi =" << std::endl;
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02480
02481
02482
02483
02484
02485
02486
02487
02488
02489
02490
02491
02492
02493
02494
02495
02496
02497
02498
02499
02500
02501
02502
02503
02504
02505
02506
02507
02508
02509
02510
02511
02512
02513
02514
02515
02516
02517
02518
02519
02520
02521
02522
02523
02524
02525
02526
02527
02528
02529
02530
02531
02532
02533
02534
02535
02536
02537
02538
02539
02540
02541
02542
02543
02544
02545
02546
02547
02548
02549
02550
02551
02552
02553
02554
02555
02556
02557
02558
02559
02560
02561
02562
02563
02564
02565
02566
02567
02568
02569
02570
02571
02572
02573
02574
02575
02576
02577
02578
02579
02580
02581
02582
02583
02584
02585
02586
02587
02588
02589
02590
02591
02592
02593
02594
02595
02596
02597
02598
02599
02600
02601
02602
02603
02604
02605
02606
02607
02608
02609
02610
02611
02612
02613
02614
02615
02616
02617
02618
02619
02620
02621
02622
02623
02624
02625
02626
02627
02628
02629
02630
02631
02632
02633
02634
02635
02636
02637
02638
02639
02640
02641
02642
02643
02644
02645
02646
02647
02648
02649
02650
02651
02652
02653
02654
02655
02656
02657
02658
02659
02660
02661
02662
02663
02664
02665
02666
02667
02668
02669
02670
02671
02672
02673
02674
02675
02676
02677
02678
02679
02680
02681
02682
02683
02684
02685
02686
02687
02688
02689
02690
02691
02692
02693
02694
02695
02696
02697
02698
02699
02700
02701
02702
02703
02704
02705
02706
02707
02708
02709
02710
02711
02712
02713
02714
02715
02716
02717
02718
02719
02720
02721
02722
02723
02724
02725
02726
02727
02728
02729
02730
02731
02732
02733
02734
02735
02736
02737
02738
02739
02740
02741
02742
02743
02744
02745
02746
02747
02748
02749
02750
02751
02752
02753
02754
02755
02756
02757
02758
02759
02760
02761
02762
02763
02764
02765
02766
02767
02768
02769
02770
02771
02772
02773
02774
02775
02776
02777
02778
02779
02780
02781
02782
02783
02784
02785
02786
02787
02788
02789
02790
02791
02792
02793
02794
02795
02796
02797
02798
02799
02800
02801
02802
02803
02804
02805
02806
02807
02808
02809
02810
02811
02812
02813
02814
02815
02816
02817
02818
02819
02820
02821
02822
02823
02824
02825
02826
02827
02828
02829
02830
02831
02832
02833
02834
02835
02836
02837
02838
02839
02840
02841
02842
02843
02844
02845
02846
02847
02848
02849
02850
02851
02852
02853
02854
02855
02856
02857
02858
02859
02860
02861
02862
02863
02864
02865
02866
02867
02868
02869
02870
02871
02872
02873
02874
02875
02876
02877
02878
02879
02880
02881
02882
02883
02884
02885
02886
02887
02888
02889
02890
02891
02892
02893
02894
02895
02896
02897
02898
02899
02900
02901
02902
02903
02904
02905
02906
02907
02908
02909
02910
02911
02912
02913
02914
02915
02916
02917
02918
02919
02920
02921
02922
02923
02924
02925
02926
02927
02928
02929
02930
02931
02932
02933
02934
02935
02936
02937
02938
02939
02940
02941
02942
02943
02944
02945
02946
02947
02948
02949
02950
02951
02952
02953
02954
02955
02956
02957
02958
02959
02960
02961
02962
02963
02964
02965
02966
02967
02968
02969
02970
```

```

01313     std::cout << phi << std::endl;
01314 #endif
01315
01317
01318     mtk::Real *lamed{}; // Used to build big lambda.
01319
01320     try {
01321         lamed = new mtk::Real[num_bndy_approxxs_ - 1];
01322     } catch (std::bad_alloc &memory_allocation_exception) {
01323         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01324         std::endl;
01325         std::cerr << memory_allocation_exception.what() << std::endl;
01326     }
01327     memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approxxs_ - 1));
01328
01329     for (auto ii = 0; ii < num_bndy_approxxs_ - 1; ++ii) {
01330         lamed[ii] = hh[ii + order_accuracy_ + 1];
01331     }
01332
01333 #if MTK_VERBOSE_LEVEL > 3
01334     std::cout << "lamed =" << std::endl;
01335     for (auto ii = 0; ii < num_bndy_approxxs_ - 1; ++ii) {
01336         std::cout << std::setw(12) << lamed[ii] << std::endl;
01337     }
01338     std::cout << std::endl;
01339 #endif
01340
01341     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01342         mtk::Real temp = mtk::kZero;
01343         for (auto jj = 0; jj < num_bndy_approxxs_ - 1; ++jj) {
01344             temp = temp +
01345                 lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01346         }
01347         hh[ii] = hh[ii] - temp;
01348     }
01349
01350 #if MTK_VERBOSE_LEVEL > 3
01351     std::cout << "big_lambda =" << std::endl;
01352     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01353         std::cout << std::setw(12) << hh[ii] << std::endl;
01354     }
01355     std::cout << std::endl;
01356 #endif
01357
01358
01359 #ifdef MTK_VERBOSE_WEIGHTS
01360     int copy_result{1};
01361 #else
01362     int copy_result{};
01363 #endif
01364
01365     int minrow_{std::numeric_limits<int>::infinity()};
01366
01367     mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01368         order_accuracy_)};
01369     mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01370
01371     mtk::Real normerr_; // Norm of the error for the solution on each row.
01372
01373 #ifdef MTK_VERBOSE_WEIGHTS
01374     std::ofstream table("grad_1d_" + std::to_string(order_accuracy_) +
01375         "_weights.tex");
01376
01377     table << "\\begin{tabular}{c|c";
01378     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01379         table << 'c';
01380     }
01381     table << ":c}\\n\\toprule\\nRow & ";
01382     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01383         table << "$ q_{\\" + std::to_string(ii) + "}$ &";
01384     }
01385     table << " Relative error \\\\n\\midrule\\n";
01386 #endif
01387
01388     for (auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01389         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01390             data(),
01391                                         order_accuracy_ + 1,
01392                                         order_accuracy_,
01393                                         order_accuracy_,
01394                                         hh,

```

```

01394                                         weights_cbs_,
01395                                         row_,
01396                                         mimetic_threshold_,
01397                                         copy_result);
01398     mtk::Real aux{normerr_/norm};
01399
01400 #if MTK_VERBOSE_LEVEL > 2
01401 std::cout << "Relative norm: " << aux << " " << std::endl;
01402 std::cout << std::endl;
01403 #endif
01404
01405 num_feasible_sols_ = num_feasible_sols_
01406     (int) (normerr_ != std::numeric_limits<mtk::Real>::infinity());
01407
01408 if (aux < minnorm) {
01409     minnorm = aux;
01410     minrow_= row_;
01411 }
01412
01413 #ifdef MTK_VERBOSE_WEIGHTS
01414 table << std::to_string(row_ + 1) << " & ";
01415 if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01416     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01417         table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01418     }
01419     table << std::to_string(aux) << " \\\\" << std::endl;
01420 } else {
01421     table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
01422         "}{c}{}\\emptyset\$" & ",";
01423     table << " - \\\\" << std::endl;
01424 }
01425 #endif
01426 }
01427
01428 #ifdef MTK_VERBOSE_WEIGHTS
01429 table << "\\midrule" << std::endl;
01430 table << "CRS weights:";
01431 for (int ii = 1; ii <= order_accuracy_; ++ii) {
01432     table << " & " << std::to_string(weights_crs_[ii - 1]);
01433 }
01434 table << " & - \\\\"n\\bottomrule\\end{tabular}" << std::endl;
01435 table.close();
01436 #endif
01437
01438 #if MTK_VERBOSE_LEVEL > 3
01439 std::cout << "weights_CBSA + lambda (after brute force search):" <<
01440     std::endl;
01441 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01442     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01443 }
01444 std::cout << std::endl;
01445 #endif
01446
01447
01448 // After we know which row yields the smallest relative norm that row is
01449 // chosen to be the objective function and the result of the optimizer is
01450 // chosen to be the new weights_.
01451
01452
01453 #if MTK_VERBOSE_LEVEL > 2
01454 std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01455     minrow_ + 1 << std::endl;
01456 std::cout << std::endl;
01457 #endif
01458
01459 copy_result = 1;
01460 normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01461 data(),
01462                                         order_accuracy_ + 1,
01463                                         order_accuracy_,
01464                                         order_accuracy_,
01465                                         hh,
01466                                         weights_cbs_,
01467                                         minrow_,
01468                                         mimetic_threshold_,
01469                                         copy_result);
01470 mtk::Real aux_{normerr_/norm};
01471 #if MTK_VERBOSE_LEVEL > 2
01472 std::cout << "Relative norm: " << aux_ << std::endl;
01473 std::cout << std::endl;
01474 #endif

```

```

01475     delete [] lamed;
01476     lamed = nullptr;
01477 }
01478
01479 delete [] hh;
01480 hh = nullptr;
01481
01482 return true;
01483 }
01484
01485 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01486
01487 #if MTK_VERBOSE_LEVEL > 3
01488 std::cout << "weights_* + lambda =" << std::endl;
01489 for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01490     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01491 }
01492 std::cout << std::endl;
01493 #endif
01494
01495
01496 mtk::Real *lambda{}; // Collection of bottom values from weights_.
01497
01498 try {
01499     lambda = new mtk::Real[dim_null_];
01500 } catch (std::bad_alloc &memory_allocation_exception) {
01501     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01502         std::endl;
01503     std::cerr << memory_allocation_exception.what() << std::endl;
01504 }
01505 memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01506
01507 for (auto ii = 0; ii < dim_null_; ++ii) {
01508     lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01509 }
01510
01511 #if MTK_VERBOSE_LEVEL > 3
01512 std::cout << "lambda =" << std::endl;
01513 for (auto ii = 0; ii < dim_null_; ++ii) {
01514     std::cout << std::setw(12) << lambda[ii] << std::endl;
01515 }
01516 std::cout << std::endl;
01517 #endif
01518
01519
01520 mtk::Real *alpha{}; // Collection of alpha values.
01521
01522 try {
01523     alpha = new mtk::Real[dim_null_];
01524 } catch (std::bad_alloc &memory_allocation_exception) {
01525     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01526         std::endl;
01527     std::cerr << memory_allocation_exception.what() << std::endl;
01528 }
01529 memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01530
01531 for (auto ii = 0; ii < dim_null_; ++ii) {
01532     alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01533 }
01534
01535 #if MTK_VERBOSE_LEVEL > 3
01536 std::cout << "alpha =" << std::endl;
01537 for (auto ii = 0; ii < dim_null_; ++ii) {
01538     std::cout << std::setw(12) << alpha[ii] << std::endl;
01539 }
01540 std::cout << std::endl;
01541 #endif
01542
01543
01544
01545 try {
01546     mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxxs_];
01547 } catch (std::bad_alloc &memory_allocation_exception) {
01548     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01549         std::endl;
01550     std::cerr << memory_allocation_exception.what() << std::endl;
01551 }
01552 memset(mim_bndy_,
01553     mtk::kZero,
01554     sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxxs_);
01555
01556 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
020010
020011
020012
020013
020014
020015
020016
020017
020018
020019
020020
020021
020022
020023
020024
020025
020026
020027
020028
020029
020030
020031
020032
020033
020034
020035
020036
020037
020038
020039
020040
020041
020042
020043
020044
020045
020046
020047
020048
020049
020050
020051
020052
020053
020054
020055
020056
020057
020058
020059
020060
020061
020062
020063
020064
020065
020066
020067
020068
020069
020070
020071
020072
020073
020074
020075
020076
020077
020078
020079
020080
020081
020082
020083
020084
020085
020086
020087
020088
020089
020090
020091
020092
020093
020094
020095
020096
020097
020098
020099
0200100
0200101
0200102
0200103
0200104
0200105
0200106
0200107
0200108
0200109
0200110
0200111
0200112
0200113
0200114
0200115
0200116
0200117
0200118
0200119
0200120
0200121
0200122
0200123
0200124
0200125
0200126
0200127
0200128
0200129
0200130
0200131
0200132
0200133
0200134
0200135
0200136
0200137
0200138
0200139
0200140
0200141
0200142
0200143
0200144
0200145
0200146
0200147
0200148
0200149
0200150
0200151
0200152
0200153
0200154
0200155
0200156
0200157
0200158
0200159
0200160
0200161
0200162
0200163
0200164
0200165
0200166
0200167
0200168
0200169
0200170
0200171
0200172
0200173
0200174
0200175
0200176
0200177
0200178
0200179
0200180
0200181
0200182
0200183
0200184
0200185
0200186
0200187
0200188
0200189
0200190
0200191
0200192
0200193
0200194
0200195
0200196
0200197
0200198
0200199
0200200
0200201
0200202
0200203
0200204
0200205
0200206
0200207
0200208
0200209
0200210
0200211
0200212
0200213
0200214
0200215
0200216
0200217
0200218
0200219
0200220
0200221
0200222
0200223
0200224
0200225
0200226
0200227
0200228
0200229
0200230
0200231
0200232
0200233
0200234
0200235
0200236
0200237
0200238
0200239
0200240
0200241
0200242
0200243
0200244
0200245
0200246
0200247
0200248
0200249
0200250
0200251
0200252
0200253
0200254
0200255
0200256
0200257
0200258
0200259
0200260
0200261
0200262
0200263
0200264
0200265
0200266
0200267
0200268
0200269
0200270
0200271
0200272
0200273
0200274
0200275
0200276
0200277
0200278
0200279
0200280
0200281
0200282
0200283
0200284
0200285
0200286
0200287
0200288
0200289
0200290
0200291
0200292
0200293
0200294
0200295
0200296
0200297
0200298
0200299
0200300
0200301
0200302
0200303
0200304
0200305
0200306
0200307
0200308
0200309
0200310
0200311
0200312
0200313
0200314
0200315
0200316
0200317
0200318
0200319
0200320
0200321
0200322
0200323
0200324
0200325
0200326
0200327
0200328
0200329
0200330
0200331
0200332
0200333
0200334
0200335
0200336
0200337
0200338
0200339
0200340
0200341
0200342
0200343
0200344
0200345
0200346
0200347
0200348
0200349
0200350
0200351
0200352
0200353
0200354
0200355
0200356
0200357
0200358
0200359
0200360
0200361
0200362
0200363
0200364
0200365
0200366
0200367
0200368
0200369
0200370
0200371
0200372
0200373
0200374
0200375
0200376
0200377
0200378
0200379
0200380
0200381
0200382
0200383
0200384
0200385
0200386
0200387
0200388
0200389
0200390
0200391
0200392
0200393
0200394
0200395
0200396
0200397
0200398
0200399
0200400
0200401
0200402
0200403
0200404
0200405
0200406
0200407
0200408
0200409
0200410
0200411
0200412
0200413
0200414
0200415
0200416
0200417
0200418
0200419
0200420
0200421
0200422
0200423
0200424
0200425
0200426
0200427
0200428
0200429
0200430
0200431
0200432
0200433
0200434
0200435
0200436
0200437
0200438
0200439
0200440
0200441
0200442
0200443
0200444
0200445
0200446
0200447
0200448
0200449
0200450
0200451
0200452
0200453
0200454
0200455
0200456
0200457
0200458
0200459
0200460
0200461
0200462
0200463
0200464
0200465
0200466
0200467
0200468
0200469
0200470
0200471
0200472
0200473
0200474
0200475
0200476
0200477
0200478
0200479
0200480
0200481
0200482
0200483
0200484
0200485
0200486
0200487
0200488
0200489
0200490
0200491
0200492
0200493
0200494
0200495
0200496
0200497
0200498
0200499
0200500
0200501
0200502
0200503
0200504
0200505
0200506
0200507
0200508
0200509
0200510
0200511
0200512
0200513
0200514
0200515
0200516
0200517
0200518
0200519
0200520
0200521
0200522
0200523
0200524
0200525
0200526
0200527
0200528
0200529
0200530
0200531
0200532
0200533
0200534
0200535
0200536
0200537
0200538
0200539
0200540
0200541
0200542
0200543
0200544
0200545
0200546
0200547
0200548
0200549
0200550
0200551
0200552
0200553
0200554
0200555
0200556
0200557
0200558
0200559
0200560
0200561
0200562
0200563
0200564
0200565
0200566
0200567
0200568
0200569
0200570
0200571
0200572
0200573
0200574
0200575
0200576
0200577
0200578
0200579
0200580
0200581
0200582
0200583
0200584
0200585
0200586
0200587
0200588
0200589
0200590
0200591
0200592
0200593
0200594
0200595
0200596
0200597
0200598
0200599
0200600
0200601
0200602
0200603
0200604
0200605
0200606
0200607
0200608
0200609
0200610
0200611
0200612
0200613
0200614
0200615
0200616
0200617
0200618
0200619
0200620
0200621
0200622
0200623
0200624
0200625
0200626
0200627
0200628
0200629
0200630
0200631
0200632
0200633
0200634
0200635
0200636
0200637
0200638
0200639
0200640
0200641
0200642
0200643
0200644
0200645
0200646
0200647
0200648
0200649
0200650
0200651
0200652
0200653
0200654
0200655
0200656
0200657
0200658
0200659
0200660
0200661
0200662
0200663
0200664
0200665
0200666
0200667
0200668
0200669
0200670
0200671
0200672
0200673
0200674
0200675
0200676
0200677
0200678
0200679
0200680
0200681
0200682
0200683
0200684
0200685
0200686
0200687
0200688
0200689
0200690
0200691
0200692
0200693
0200694
0200695
0200696
0200697
0200698
0200699
0200700
0200701
0200702
0200703
0200704
0200705
0200706
0200707
0200708
0200709
0200710
0200711
0200712
0200713
0200714
0200715
0200716
0200717
0200718
0200719
0200720
0200721
0200722
0200723
0200724
0200725
0200726
0200727
0200728
0200729
0200730
0200731
0200732
0200733
0200734
0200735
0200736
0200737
0200738
0200739
0200740
0200741
0200742
0200743
0200744
0200745
0200746
0200747
0200748
0200749
0200750
0200751
0200752
0200753
0200754
0200755
0200756
0200757
0200758
0200759
0200760
0200761
0200762
0200763
0200764
0200765
0200766
0200767
0200768
0200769
0200770
0200771
0200772
0200773
0200774
0200775
0200776
0200777
0200778
0200779
0200780
0200781
0200782
0200783
0200784
0200785
0200786
0200787
0200788
0200789
0200790
0200791
0200792
0200793
0200794
0200795
0200796
0200797
0200798
0200799
0200800
0200801
0200802
0200803
0200804
0200805
0200806
0200807
0200808
0200809
0200810
0200811
0200812
0200813
0200814
0200815
0200816
0200817
0200818
0200819
0200820
0200821
0200822
0200823
0200824
0200825
0200826
0200827
0200828
0200829
0200830
0200831
0200832
0200833
0200834
0200835
0200836
0200837
0200838
0200839
0200840
0200841
0200842
0200843
0200844
0200845
0200846
0200847
0200848
0200849
0200850
0200851
0200852
0200853
0200854
0200855
0200856
0200857
0200858
0200859
0200860
0200861
0200862
0200863
0200864
0200865
0200866
0200867
0200868
0200869
0200870
0200871
0200872
0200873
0200874
0200875
0200876
0200877
0200878
0200879
0200880
0200881
0200882
0200883
0200884
0200885
0200886
0200887
0200888
0200889
0200890
0200891
0200892
0200893
0200894
0200895
0200896
0200897
0200898
0200899
0200900
0200901
0200902
0200903
0200904
0200905
0200906
0200907
0200908
0200909
0200910
0200911
0200912
0200913
0200914
0200915
0200916
0200917
0200918
0200919
0200920
0200921
0200922
0200923
0200924
0200925
0200926
0200927
0200928
0200929
0200930
0200931
0200932
0200933
0200934
0200935
0200936
0200937
0200938
0200939
0200940
0200941
0200942
0200943
0200944
0200945
0200946
0200947
0200948
0200949
0200950
0200951
0200952
0200953
0200954
0200955
0200956
0200957
0200958
0200959
0200960
0200961
0200962
0200963
0200964
0200965
0200966
0200967
0200968
0200969
0200970
0200971
0200972
0200973
0200974
0200975
0200976
0200977
0200978
0200979
0200980
0200981
0200982
0200983
0200984
0200985
0200986
0200987
0200988
0200989
0200990
0200991
0200992
0200993
0200994
0200995
0200996
0200997
0200998
0200999
0200100
0200101
0200102
0200103
0200104
0200105
0200106
0200107
0200108
0200109
0200110
0200111
0200112
0200113
0200114
0200115
0200116
0200117
0200118
0200119
0200120
0200121
0200122
0200123
0200124
0200125
0200126
0200127
0200128
0200129
0200130
0200131
0200132
0200133
0200134
0200135
0200136
0200137
0200138
0200139
0200140
0200141
0200142
0200143
0200144
0200145
0200146
0200147
0200148
0200149
0200150
0200151
0200152
0200153
0200154
0200155
0200156
0200157
0200158
0200159
0200160
0200161
0200162
0200163
0200164
0200165
0200166
0200167
```

```

01559     for (auto jj = 0; jj < (num_bndy_approxxs_ - 1); ++jj) {
01560         mim_bndy_[ii*num_bndy_approxxs_ + jj] =
01561             prem_apps_[ii*num_bndy_approxxs_ + jj] +
01562                 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01563     }
01564 }
01565
01566 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01567     mim_bndy_[ii*num_bndy_approxxs_ + (num_bndy_approxxs_ - 1)] =
01568         prem_apps_[ii*num_bndy_approxxs_ + (num_bndy_approxxs_ - 1)];
01569 }
01570
01571 #if MTK_VERBOSE_LEVEL > 4
01572 std::cout << "Collection of mimetic approximations:" << std::endl;
01573 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01574     for (auto jj = 0; jj < num_bndy_approxxs_; ++jj) {
01575         std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxxs_ + jj];
01576     }
01577     std::cout << std::endl;
01578 }
01579 std::cout << std::endl;
01580 #endif
01581
01582
01583 for (auto ii = 0; ii < num_bndy_approxxs_; ++ii) {
01584     sums_rows_mim_bndy_.push_back(mtk::kZero);
01585
01586
01587
01588
01589 for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01590     sums_rows_mim_bndy_[ii] += mim_bndy_[jj*num_bndy_approxxs_ + ii];
01591 }
01592
01593 }
01594
01595 mimetic_measure_ = *std::max_element(sums_rows_mim_bndy_.begin(),
01596                                         sums_rows_mim_bndy_.end());
01597
01598 #if MTK_VERBOSE_LEVEL > 3
01599 std::cout << "Row-wise sum of mimetic approximations:" << std::endl;
01600 for (auto ii = 0; ii < num_bndy_approxxs_; ++ii) {
01601     std::cout << std::setw(13) << sums_rows_mim_bndy_[ii];
01602 }
01603 std::cout << std::endl;
01604 std::cout << std::endl;
01605 #endif
01606
01607 delete[] lambda;
01608 lambda = nullptr;
01609
01610 delete[] alpha;
01611 alpha = nullptr;
01612
01613 return true;
01614 }
01615
01616 bool mtk::Grad1D::AssembleOperator(void) {
01617
01618     // The output array will have this form:
01619     // 1. The first entry of the array will contain the used order kk.
01620     // 2. The second entry of the array will contain the collection of
01621     // approximating coefficients for the interior of the grid.
01622     // 3. The third entry will contain a collection of weights.
01623     // 4. The next dim_null - 1 entries will contain the collections of
01624     // approximating coefficients for the west boundary of the grid.
01625
01626     gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01627         num_bndy_approxxs_*num_bndy_coeffs_;
01628
01629 #if MTK_VERBOSE_LEVEL > 2
01630 std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01631 #endif
01632
01633 try {
01634     gradient_ = new mtk::Real[gradient_length_];
01635 } catch (std::bad_alloc &memory_allocation_exception) {
01636     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01637     std::endl;
01638     std::cerr << memory_allocation_exception.what() << std::endl;
01639 }
01640 memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);

```

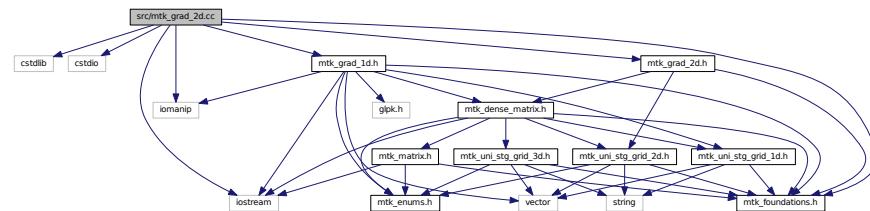
```

01641
01643
01644     gradient_[0] = order_accuracy_;
01645
01648
01649     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01650         gradient_[ii + 1] = coeffs_interior_[ii];
01651     }
01652
01654
01655     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01656         gradient_[order_accuracy_ + 1 + ii] = weights_cbs_[ii];
01657     }
01658
01659
01660     int offset{2*order_accuracy_ + 1};
01661
01662     int aux {}; // Auxiliary variable.
01663
01664
01665     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01666         for (auto ii = 0; ii < num_bndy_approxs_; ii++) {
01667             for (auto jj = 0; jj < num_bndy_coeffs_; jj++) {
01668                 gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01669                 aux++;
01670             }
01671         }
01672     }
01673 } else {
01674     gradient_[offset + 0] = prem_apps_[0];
01675     gradient_[offset + 1] = prem_apps_[1];
01676     gradient_[offset + 2] = prem_apps_[2];
01677 }
01678
01679 #if MTK_VERBOSE_LEVEL > 1
01680 std::cout << "1D " << order_accuracy_ << "-order grad built!" << std::endl;
01681 std::cout << std::endl;
01682 #endif
01683
01684     return true;
01685 }
```

18.105 src/mtk_grad_2d.cc File Reference

Implements the class Grad2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_2d.h"
Include dependency graph for mtk_grad_2d.cc:
```



18.105.1 Detailed Description

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

Author

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

Definition in file [mtk_grad_2d.cc](#).

18.106 mtk_grad_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_2d.h"
00066
00067 mtk::Grad2D::Grad2D():
00068     order_accuracy_(),
00069     mimetic_threshold_() {}

```

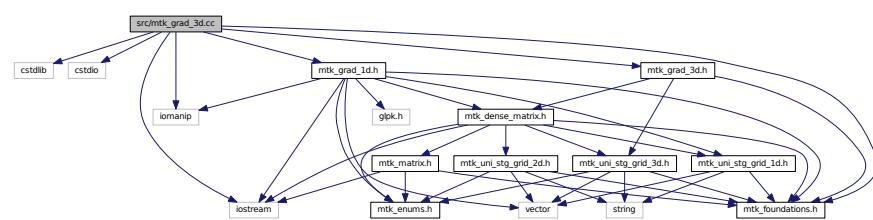
```

00070
00071 mtk::Grad2D::Grad2D(const Grad2D &grad):
00072     order_accuracy_(grad.order_accuracy_),
00073     mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad2D::~Grad2D() {}
00076
00077 bool mtk::Grad2D::ConstructGrad2D(const
00078     mtk::UniStgGrid2D &grid,
00079             int order_accuracy,
00080             mtk::Real mimetic_threshold) {
00081     int num_cells_x = grid.num_cells_x();
00082     int num_cells_y = grid.num_cells_y();
00083
00084     int mx = num_cells_x + 1; // Gx vertical dimension
00085     int nx = num_cells_x + 2; // Gx horizontal dimension
00086     int my = num_cells_y + 1; // Gy vertical dimension
00087     int ny = num_cells_y + 2; // Gy horizontal dimension
00088
00089     mtk::Grad1D grad;
00090
00091     bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00092
00093 #ifdef MTK_PERFORM_PREVENTIONS
00094     if (!info) {
00095         std::cerr << "Mimetic grad could not be built." << std::endl;
00096         return info;
00097     }
00098 #endif
00099
00100     auto west = grid.west_bndy();
00101     auto east = grid.east_bndy();
00102     auto south = grid.south_bndy();
00103     auto north = grid.east_bndy();
00104
00105     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00106     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00107
00108     mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00109     mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00110
00111     bool padded{true};
00112     bool transpose{true};
00113
00114     mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00115     mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00116
00117     mtk::DenseMatrix gxy(mtk::DenseMatrix::Kron(tiy, Gx));
00118     mtk::DenseMatrix gyx(mtk::DenseMatrix::Kron(Gy, tix));
00119
00120 #if MTK_VERBOSE_LEVEL > 2
00121     std::cout << "Gx: " << mx << " by " << nx << std::endl;
00122     std::cout << "Transpose Iy: " << num_cells_y << " by " << ny << std::endl;
00123     std::cout << "Gy: " << my << " by " << ny << std::endl;
00124     std::cout << "Transpose Ix: " << num_cells_x << " by " << nx << std::endl;
00125     std::cout << "Grad 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00126     nx*ny << std::endl;
00127 #endif
00128
00129     mtk::DenseMatrix g2d(mx*num_cells_y + my*num_cells_x, nx*ny);
00130
00131     for(auto ii = 0; ii < nx*ny; ii++) {
00132         for(auto jj = 0; jj < mx*num_cells_y; jj++) {
00133             g2d.SetValue(jj, ii, gxy.GetValue(jj, ii));
00134         }
00135         for(auto kk = 0; kk < my*num_cells_x; kk++) {
00136             g2d.SetValue(kk + mx*num_cells_y, ii, gyx.GetValue(kk, ii));
00137         }
00138     }
00139
00140     gradient_ = g2d;
00141
00142     return info;
00143 }
00144
00145 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix() const {
00146
00147     return gradient_;
00148 }
```

18.107 src/mtk_grad_3d.cc File Reference

Implements the class Grad3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_3d.h"
Include dependency graph for mtk_grad_3d.cc:
```



18.107.1 Detailed Description

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

Author

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

Definition in file [mtk_grad_3d.cc](#).

18.108 mtk_grad_3d.cc

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
```

```
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_3d.h"
00066
00067 mtk::Grad3D::Grad3D():
00068     order_accuracy_(),
00069     mimetic_threshold_() {}
00070
00071 mtk::Grad3D::Grad3D(const Grad3D &grad):
00072     order_accuracy_(grad.order_accuracy_),
00073     mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad3D::~Grad3D() {}
00076
00077 bool mtk::Grad3D::ConstructGrad3D(const
00078                                     mtk::UniStgGrid3D &grid,
00079                                     int order_accuracy,
00080                                     mtk::Real mimetic_threshold) {
00081     int num_cells_x = grid.num_cells_x();
00082     int num_cells_y = grid.num_cells_y();
00083     int num_cells_z = grid.num_cells_z();
00084
00085     int mx = num_cells_x + 1; // Gx vertical dimension.
00086     int nx = num_cells_x + 2; // Gx horizontal dimension.
00087     int my = num_cells_y + 1; // Gy vertical dimension.
00088     int ny = num_cells_y + 2; // Gy horizontal dimension.
00089     int mz = num_cells_z + 1; // Gz vertical dimension.
00090     int nz = num_cells_z + 2; // Gz horizontal dimension.
00091
00092     mtk::Grad1D grad;
00093
00094     bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00095
00096 #ifdef MTK_PERFORM_PREVENTIONS
00097     if (!info) {
00098         std::cerr << "Mimetic grad could not be built." << std::endl;
00099         return info;
00100     }
00101 #endif
00102
00103     auto west = grid.west_bndy();
00104     auto east = grid.east_bndy();
00105     auto south = grid.south_bndy();
00106     auto north = grid.north_bndy();
00107     auto bottom = grid.bottom_bndy();
00108     auto top = grid.top_bndy();
00109
00110     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00111     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00112     mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z);
```

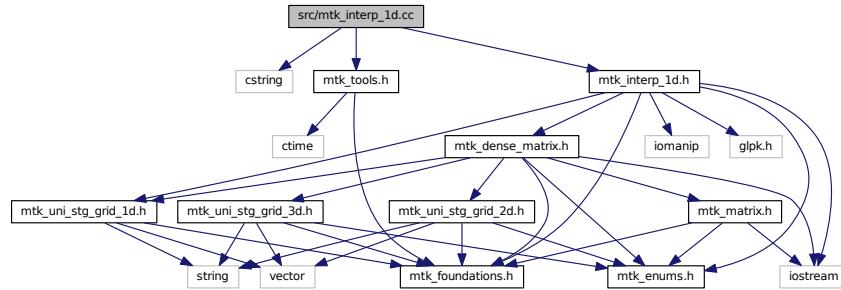
```

00113
00114     mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00115     mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00116     mtk::DenseMatrix Gz(grad.ReturnAsDenseMatrix(grid_z));
00117
00118     bool padded{true};
00119     bool transpose{true};
00120
00121     mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00122     mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00123     mtk::DenseMatrix tiz(num_cells_z, padded, transpose);
00124
00125
00126     mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(tiz, tiy));
00127     mtk::DenseMatrix gx(mtk::DenseMatrix::Kron(aux1, Gx));
00128
00129
00130
00131     mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(tiz, Gy));
00132     mtk::DenseMatrix gy(mtk::DenseMatrix::Kron(aux2, tix));
00133
00134
00135
00136     mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Gz, tiy));
00137     mtk::DenseMatrix gz(mtk::DenseMatrix::Kron(aux3, tix));
00138
00139
00140 #if MTK_VERBOSE_LEVEL > 2
00141     std::cout << "Gx: " << mx << " by " << nx << std::endl;
00142     std::cout << "Transpose Ix: " << num_cells_x << " by " << nx << std::endl;
00143     std::cout << "Gy: " << my << " by " << ny << std::endl;
00144     std::cout << "Transpose Iy: " << num_cells_y << " by " << ny << std::endl;
00145     std::cout << "Gz: " << mz << " by " << nz << std::endl;
00146     std::cout << "Transpose Iz: " << num_cells_z << " by " << nz << std::endl;
00147 #endif
00148
00149
00150     int total_rows{mx*num_cells_y*num_cells_z +
00151                 num_cells_x*my*num_cells_z +
00152                 num_cells_x*num_cells_y*mz};
00153     int total_cols{nx*ny*nz};
00154
00155
00156 #if MTK_VERBOSE_LEVEL > 2
00157     std::cout << "Grad 3D: " << total_rows << " by " << total_cols << std::endl;
00158 #endif
00159
00160     mtk::DenseMatrix g3d(total_rows, total_cols);
00161
00162     for(auto ii = 0; ii < total_cols; ii++) {
00163         for(auto jj = 0; jj < mx*num_cells_y*num_cells_z; jj++) {
00164             g3d.SetValue(jj,ii, gx.GetValue(jj,ii));
00165         }
00166
00167         int offset = mx*num_cells_y*num_cells_z;
00168
00169         for(auto kk = 0; kk < num_cells_x*my*num_cells_z; kk++) {
00170             g3d.SetValue(kk + offset, ii, gy.GetValue(kk,ii));
00171         }
00172
00173         offset += num_cells_x*my*num_cells_z;
00174
00175         for(auto ll = 0; ll < num_cells_x*num_cells_y*mz; ll++) {
00176             g3d.SetValue(ll + offset, ii, gz.GetValue(ll,ii));
00177         }
00178     }
00179
00180     gradient_ = g3d;
00181
00182     return info;
00183 }
00184
00185 mtk::DenseMatrix mtk::Grad3D::ReturnAsDenseMatrix() const {
00186     return gradient_;
00187 }
00188 }
```

18.109 src/mtk_interp_1d.cc File Reference

Includes the implementation of the class `Interp1D`.

```
#include <cstring>
#include "mtk_tools.h"
#include "mtk_interp_1d.h"
Include dependency graph for mtk_interp_1d.cc:
```



Namespaces

- `mtk`

Mimetic Methods Toolkit namespace.

Functions

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

18.109.1 Detailed Description

This class implements a 1D interpolation operator.

Author

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

Definition in file [mtk_interp_1d.cc](#).

18.110 mtk_interp_1d.cc

```
00001
00012 /*
00013 Copyright (C) 2016, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
```

```

00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #include <cstring>
00059
00060 #include "mtk_tools.h"
00061
00062 #include "mtk_interp_1d.h"
00063
00064 namespace mtk {
00065
00066 std::ostream& operator <<(std::ostream &stream, mtk::Interp1D &in) {
00067
00068
00069     stream << "coeffs_interior_[1:" << in.order_accuracy_ << "] = ";
00070     for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00071         stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00072     }
00073
00074     stream << std::endl;
00075
00076     return stream;
00077 }
00078 }
00079
00080 mtk::Interp1D::Interp1D():
00081     dir_interp_(mtk::DirInterp::SCALAR_TO_VECTOR),
00082     order_accuracy_(mtk::kDefaultOrderAccuracy),
00083     coeffs_interior_(nullptr)
00084
00085 mtk::Interp1D::Interp1D(const Interp1D &interp):
00086     dir_interp_(interp.dir_interp_),
00087     order_accuracy_(interp.order_accuracy_),
00088     coeffs_interior_(interp.coeffs_interior_) {}
00089
00090 mtk::Interp1D::~Interp1D() {
00091
00092     delete[] coeffs_interior_;
00093     coeffs_interior_ = nullptr;
00094 }
00095
00096 bool mtk::Interp1D::ConstructInterp1D(int order_accuracy,
00097     mtk::DirInterp dir) {
00098
00099     #if MTK_PERFORM_PREVENTIONS
00100     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00101     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00102     mtk::Tools::Prevent(dir < mtk::DirInterp::SCALAR_TO_VECTOR
00103     &&
00104             dir > mtk::DirInterp::VECTOR_TO_SCALAR,
00105             __FILE__, __LINE__, __func__);
00106     #endif

```

```

00105
00106 #if MTK_VERBOSE_LEVEL > 2
00107 std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00108 #endif
00109
00110 order_accuracy_ = order_accuracy;
00111
00113
00114 try {
00115   coeffs_interior_ = new mtk::Real[order_accuracy_];
00116 } catch (std::bad_alloc &memory_allocation_exception) {
00117   std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00118   std::endl;
00119   std::cerr << memory_allocation_exception.what() << std::endl;
00120 }
00121 memset(coeffs_interior_,
00122     mtk::kZero,
00123     sizeof(coeffs_interior_[0])*order_accuracy_);
00124
00125 for (int ii = 0; ii < order_accuracy_; ++ii) {
00126   coeffs_interior_[ii] = mtk::kOne;
00127 }
00128
00129 return true;
00130 }
00131
00132 mtk::Real *mtk::Interp1D::coeffs_interior() const {
00133   return coeffs_interior_;
00135 }
00136
00137 mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix(
00138   const UniStgGrid1D &grid) const {
00139
00140   int nn(grid.num_cells_x()); // Number of cells on the grid.
00141
00142 #if MTK_PERFORM_PREVENTIONS
00143 mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00144 #endif
00145
00146   int gg_num_rows(); // Number of rows.
00147   int gg_num_cols(); // Number of columns.
00148
00149   if (dir_interp_ == mtk::DirInterp::SCALAR_TO_VECTOR) {
00150     gg_num_rows = nn + 1;
00151     gg_num_cols = nn + 2;
00152   } else {
00153     gg_num_rows = nn + 2;
00154     gg_num_cols = nn + 1;
00155   }
00156
00157   // Output matrix featuring sizes for gradient operators.
00158
00159   mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00160
00161   out.set_encoded_operator(
00162     mtk::EncodedOperator::INTERPOLATION);
00163
00164
00165   out.SetValue(0, 0, mtk::kOne);
00166
00167
00168
00169   for (auto ii = 1; ii < gg_num_rows - 1; ++ii) {
00170     for (auto jj = ii; jj < order_accuracy_ + ii; ++jj) {
00171       out.SetValue(ii, jj, mtk::kOne/order_accuracy_);
00172     }
00173   }
00174
00175
00176
00177   out.SetValue(gg_num_rows - 1, gg_num_cols - 1, mtk::kOne);
00178
00179   return out;
00180 }

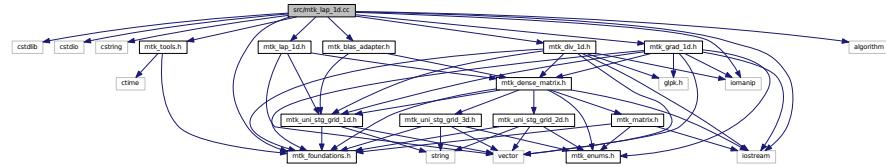
```

18.111 src/mtk_lap_1d.cc File Reference

Includes the implementation of the class Lap1D.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include "mtk_foundations.h"
#include "mtk_tools.h"
#include "mtk blas adapter.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
Include dependency graph for mtk_lap_1d.c
```

Include dependency graph for mtk_lap_1d.cc:



Namespaces

- mtk

Mimetic Methods Toolkit namespace.

Functions

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

18.111.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_1d.cc](#).

18.112 mtk_lap_1d.cc

00001
00011 /*

```
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059 #include <cstring>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include <algorithm>
00065
00066 #include "mtk_foundations.h"
00067 #include "mtk_tools.h"
00068 #include "mtk blas_adapter.h"
00069 #include "mtk_grad_1d.h"
00070 #include "mtk_div_1d.h"
00071 #include "mtk_lap_1d.h"
00072
00073 namespace mtk {
00074
00075 std::ostream& operator <<(std::ostream &stream, mtk::Lap1D &in) {
00076
00077     int output_precision{4};
00078     int output_width{8};
00079
00080
00081     stream << "Order of accuracy: " << in.laplacian_[0] << std::endl;
00082
00083
00084     stream << "Interior stencil: " << std::endl;
00085     for (auto ii = 1; ii <= (2*in.order_accuracy_ - 1); ++ii) {
00086         stream << std::setprecision(output_precision) << std::setw(output_width) <<
00087             in.laplacian_[ii] << ' ';
00088     }
00089     stream << std::endl;
00090
00091     auto offset = 1 + (2*in.order_accuracy_ - 1);
00092
00093 }
```

```

00096
00097     for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00098         stream << "Boundary row " << ii + 1 << ":" << std::endl;
00099         for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00100             stream << std::setprecision(output_precision) <<
00101                 std::setw(output_width) <<
00102                     in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj] << ' ';
00103         }
00104         stream << std::endl;
00105         stream << "Sum of elements in boundary row " << ii + 1 << ":" <<
00106             in.sums_rows_mim_bndy_[ii];
00107         stream << std::endl;
00108     }
00109
00110     return stream;
00111 }
00112 }
00113
00114 mtk::Lap1D::Lap1D():
00115     order_accuracy_(mtk::kDefaultOrderAccuracy),
00116     laplacian_length_(),
00117     delta_(mtk::kZero),
00118     mimetic_threshold_(mtk::kDefaultMimeticThreshold),
00119     mimetic_measure_(mtk::kZero),
00120     sums_rows_mim_bndy_({})
00121
00122 mtk::Lap1D::Lap1D(const Lap1D &lap):
00123     order_accuracy_(lap.order_accuracy_),
00124     laplacian_length_(lap.laplacian_length_),
00125     delta_(lap.delta_),
00126     mimetic_threshold_(lap.mimetic_threshold_),
00127     mimetic_measure_(lap.mimetic_measure_),
00128     sums_rows_mim_bndy_(lap.sums_rows_mim_bndy_) {}
00129
00130 mtk::Lap1D::~Lap1D() {
00131
00132     delete [] laplacian_;
00133     laplacian_ = nullptr;
00134 }
00135
00136 int mtk::Lap1D::order_accuracy() const {
00137
00138     return order_accuracy_;
00139 }
00140
00141 mtk::Real mtk::Lap1D::mimetic_threshold() const {
00142
00143     return mimetic_threshold_;
00144 }
00145
00146 mtk::Real mtk::Lap1D::delta() const {
00147
00148     return delta_;
00149 }
00150
00151 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00152                                     mtk::Real mimetic_threshold) {
00153
00154 #ifdef MTK_PERFORM_PREVENTIONS
00155     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00156     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00157     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00158                         __FILE__, __LINE__, __func__);
00159
00160     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00161         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00162     }
00163
00164     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00165     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00166     #endif
00167
00168     order_accuracy_ = order_accuracy;
00169     mimetic_threshold_ = mimetic_threshold;
00170
00171     mtk::Grad1D grad; // Mimetic gradient.
00172
00173     bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00174
00175 #ifdef MTK_PERFORM_PREVENTIONS
00176     if (!info) {
00177

```

```

00178     std::cerr << "Mimetic grad could not be built." << std::endl;
00179     return false;
00180 }
00181 #endif
00182
00184
00185 mtk::Div1D div; // Mimetic divergence.
00186
00187 info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00188
00189 #ifdef MTK_PERFORM_PREVENTIONS
00190 if (!info) {
00191     std::cerr << "Mimetic div could not be built." << std::endl;
00192     return false;
00193 }
00194 #endif
00195
00196
00197 // Since these are mimetic operator, we must multiply the matrices arising
00198 // from both the divergence and the Laplacian, in order to get the
00199 // approximating coefficients for the Laplacian operator.
00200
00201 // However, we must choose a grid that implied a step size of 1, so to get
00202 // the approximating coefficients, without being affected from the
00203 // normalization with respect to the grid (dimensionless).
00204
00205 // Also, the grid must be of the minimum size to support the requested order
00206 // of accuracy. We must please the divergence for this!
00207
00208 mtk::UniStgGrid1D aux(mtk::kZero,
00209                         (mtk::Real) 3*order_accuracy_ - 1,
00210                         3*order_accuracy_ - 1);
00211
00212
00213 #if MTK_VERBOSE_LEVEL > 2
00214 std::cout << "aux =" << std::endl;
00215 std::cout << aux << std::endl;
00216 std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00217 std::cout << std::endl;
00218 #endif
00219
00220 mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00221
00222 #if MTK_VERBOSE_LEVEL > 4
00223 std::cout << "grad_m =" << std::endl;
00224 std::cout << grad_m << std::endl;
00225 #endif
00226
00227 mtk::UniStgGrid1D aux2(mtk::kZero,
00228                         (mtk::Real) 3*order_accuracy_ - 1,
00229                         3*order_accuracy_ - 1,
00230                         mtk::FieldNature::VECTOR);
00231
00232 mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux2));
00233
00234 #if MTK_VERBOSE_LEVEL > 4
00235 std::cout << "div_m =" << std::endl;
00236 std::cout << div_m << std::endl;
00237 #endif
00238
00239
00240
00241 mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00242
00243 lap.set_encoded_operator(mtk::EncodedOperator::LAPLACIAN
00244 );
00245
00246 lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00247
00248 #if MTK_VERBOSE_LEVEL > 4
00249 std::cout << "lap =" << std::endl;
00250 std::cout << lap << std::endl;
00251 #endif
00252
00253
00254
00255
00256
00257 // The output array will have this form:
00258 // 1. The first entry of the array will contain the used order kk.
00259 // 2. The second entry of the array will contain the collection of
00260 // approximating coefficients for the interior of the grid.
00261 // 3. The next entries will contain the collections of approximating
00262 // coefficients for the west boundary of the grid.
00263
00264

```

```

00265     laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00266         (order_accuracy_ - 1)*(2*order_accuracy_);
00267
00268 #if MTK_VERBOSE_LEVEL > 2
00269 std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00270 std::cout << std::endl;
00271 #endif
00272
00273 try {
00274     laplacian_ = new mtk::Real[laplacian_length_];
00275 } catch (std::bad_alloc &memory_allocation_exception) {
00276     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00277         std::endl;
00278     std::cerr << memory_allocation_exception.what() << std::endl;
00279 }
00280 memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00281
00282
00283 laplacian_[0] = order_accuracy_;
00284
00285
00286
00287 for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00288     laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00289 }
00290
00291
00292
00293 auto offset = 1 + (2*order_accuracy_ - 1);
00294
00295 for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00296     sums_rows_mim_bndy_.push_back(mtk::kZero);
00297     for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00298         register mtk::Real aux{lap.GetValue(1 + ii,jj)};
00299         laplacian_[offset + ii*(2*order_accuracy_) + jj] = aux;
00300         sums_rows_mim_bndy_[ii] += aux;
00301     }
00302 }
00303
00304 }
00305
00306 mimetic_measure_ = *std::max_element(sums_rows_mim_bndy_.begin(),
00307                                         sums_rows_mim_bndy_.end());
00308
00309 delta_ = mtk::kZero;
00310
00311 return true;
00312 }
00313
00314 std::vector<mtk::Real> mtk::Lap1D::sums_rows_mim_bndy() const {
00315
00316     return sums_rows_mim_bndy_;
00317 }
00318
00319 mtk::Real mtk::Lap1D::mimetic_measure() const {
00320
00321     return mimetic_measure_;
00322 }
00323
00324 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(
00325     const UniStgGrid1D &grid) const {
00326
00327     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00328
00329 #ifdef MTK_PERFORM_PREVENTIONS
00330     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00331     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00332 #endif
00333
00334     mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00335
00336     lap.set_encoded_operator(mtk::EncodedOperator::LAPLACIAN
00337 );
00338     delta_ = grid.delta_x();
00339
00340     mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
00341     dx^2.
00342
00343
00344     auto offset = (1 + 2*order_accuracy_ - 1);
00345
00346     for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00347         for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00348             lap.SetValue(1 + ii,

```

```

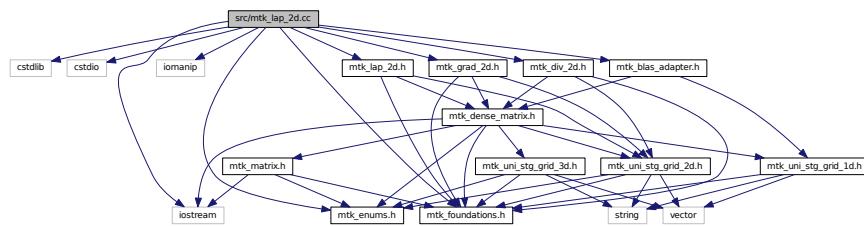
00349                     jj,
00350                     idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00351     }
00352 }
00353
00355
00356 offset = 1 + (order_accuracy_ - 1);
00357
00358 int kk{1};
00359 for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00360     int mm{1};
00361     for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00362         lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00363         mm = mm + 1;
00364     }
00365     kk = kk + 1;
00366 }
00367
00369
00370 offset = (1 + 2*order_accuracy_ - 1);
00371
00372 auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00373
00374 auto ll = 1;
00375 auto rr = 1;
00376 for (auto ii = nn; ii > aux - 1; --ii) {
00377     auto cc = 0;
00378     for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00379         lap.SetValue(ii, jj, lap.GetValue(rr, cc));
00380         ++ll;
00381         ++cc;
00382     }
00383     rr++;
00384 }
00385
00392
00393     return lap;
00394 }
00395
00396 const mtk::Real* mtk::Lap1D::data(const UniStgGrid1D &grid) const {
00397
00398     mtk::DenseMatrix tmp;
00399
00400     tmp = ReturnAsDenseMatrix(grid);
00401
00402     return tmp.data();
00403 }
```

18.113 src/mtk_lap_2d.cc File Reference

Includes the implementation of the class Lap2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtkEnums.h"
#include "mtkblas_adapter.h"
#include "mtkgrad_2d.h"
#include "mtkdiv_2d.h"
#include "mtklap_2d.h"
```

Include dependency graph for mtk_lap_2d.cc:



18.113.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_2d.cc](#).

18.114 mtk_lap_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
  
```

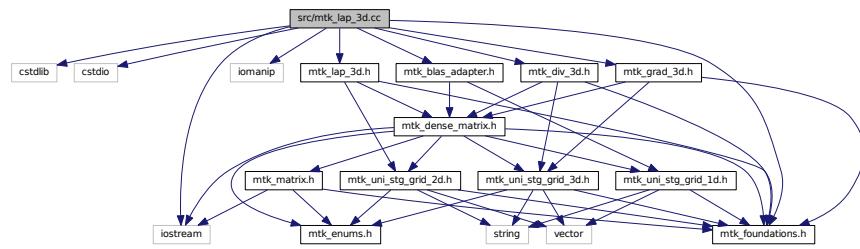
```

00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtkEnums.h"
00065 #include "mtkblas_adapter.h"
00066 #include "mtk_grad_2d.h"
00067 #include "mtk_div_2d.h"
00068 #include "mtk_lap_2d.h"
00069
00070 mtk::Lap2D::Lap2D(): order_accuracy_( ), mimetic_threshold_( ) {}
00071
00072 mtk::Lap2D::Lap2D( const Lap2D &lap):
00073     order_accuracy_( lap.order_accuracy_ ),
00074     mimetic_threshold_( lap.mimetic_threshold_ ) {}
00075
00076 mtk::Lap2D::~Lap2D() {}
00077
00078 bool mtk::Lap2D::ConstructLap2D( const
00079                                     mtk::UniStgGrid2D &grid,
00080                                     int order_accuracy,
00081                                     mtk::Real mimetic_threshold) {
00082
00083     mtk::Grad2D gg;
00084     mtk::Div2D dd;
00085
00086     bool info{gg.ConstructGrad2D( grid, order_accuracy, mimetic_threshold )};
00087
00088 #ifdef MTK_PERFORM_PREVENTIONS
00089     if ( !info ) {
00090         std::cerr << "Mimetic lap could not be built." << std::endl;
00091         return info;
00092     }
00093 #endif
00094
00095     info = dd.ConstructDiv2D( grid, order_accuracy, mimetic_threshold );
00096
00097 #ifdef MTK_PERFORM_PREVENTIONS
00098     if ( !info ) {
00099         std::cerr << "Mimetic div could not be built." << std::endl;
00100         return info;
00101     }
00102 #endif
00103
00104     mtk::DenseMatrix ggm( gg.ReturnAsDenseMatrix() );
00105     mtk::DenseMatrix ddm( dd.ReturnAsDenseMatrix() );
00106
00107     laplacian_ = mtk::BLASAdapter::RealDenseMM( ddm, ggm );
00108
00109     return info;
00110 }
00111
00112 mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix() const {
00113
00114     return laplacian_;
00115 }
00116 mtk::Real *mtk::Lap2D::data() const {
00117
00118     return laplacian_.data();
00119 }
```

18.115 src/mtk_lap_3d.cc File Reference

Includes the implementation of the class Lap3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_foundations.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"
Include dependency graph for mtk_lap_3d.cc:
```



18.115.1 Detailed Description

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

Author

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

Definition in file [mtk_lap_3d.cc](#).

18.116 mtk_lap_3d.cc

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
```

```
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdint>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_foundations.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_3d.h"
00066 #include "mtk_div_3d.h"
00067 #include "mtk_lap_3d.h"
00068
00069 mtk::UniStgGrid3D mtk::Lap3D::operator*(const
00070     mtk::UniStgGrid3D &grid) const {
00071     mtk::UniStgGrid3D out;
00072
00073     return out;
00074 }
00075
00076 mtk::Lap3D::Lap3D(): order_accuracy_(), mimetic_threshold_() {}
00077
00078 mtk::Lap3D::Lap3D(const Lap3D &lap):
00079     order_accuracy_(lap.order_accuracy_),
00080     mimetic_threshold_(lap.mimetic_threshold_) {}
00081
00082 mtk::Lap3D::~Lap3D() {}
00083
00084 bool mtk::Lap3D::ConstructLap3D(const
00085     mtk::UniStgGrid3D &grid,
00086             int order_accuracy,
00087             mtk::Real mimetic_threshold) {
00088
00089     mtk::Grad3D gg;
00090     mtk::Div3D dd;
00091
00092     bool info{gg.ConstructGrad3D(grid, order_accuracy, mimetic_threshold)};
00093
00094 #ifdef MTK_PERFORM_PREVENTIONS
00095     if (!info) {
00096         std::cerr << "Mimetic lap could not be built." << std::endl;
00097         return info;
00098     }
00099 #endif
00100
00101     info = dd.ConstructDiv3D(grid, order_accuracy, mimetic_threshold);
00102
00103 #ifdef MTK_PERFORM_PREVENTIONS
00104     if (!info) {
00105         std::cerr << "Mimetic div could not be built." << std::endl;
00106         return info;
00107     }
00108 #endif
00109
00110     mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00111     mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00112
00113     laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
```

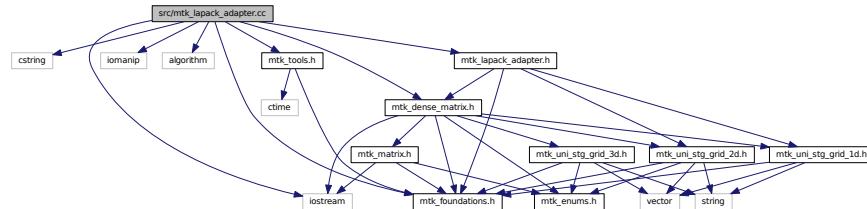
```

00113
00114     return info;
00115 }
00116
00117 mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix() const {
00118
00119     return laplacian_;
00120 }
00121
00122 mtk::Real *mtk::Lap3D::data() const {
00123
00124     return laplacian_.data();
00125 }
```

18.117 src/mtk_lapack_adapter.cc File Reference

Definition of an adapter class for the LAPACK API.

```
#include <cstring>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include "mtk_foundations.h"
#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::sgeqr_](#)_(int *m, int *n, Real *a, int *lda, Real *tau, Real *work, int *lwork, int *info)

Single-precision GEneral matrix QR Factorization.

- void [mtk::sormqr_](#)_(char *side, char *trans, int *m, int *n, int *k, Real *a, int *lda, Real *tau, Real *c, int *ldc, Real *work, int *lwork, int *info)

Single-precision Orthogonal Matrix from QR factorization.

18.117.1 Detailed Description

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

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Todo Write documentation using LaTeX.

Author

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

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

18.118 mtk_lapack_adapter.cc

```

00001
00022 /*
00023 Copyright (C) 2016, Computational Science Research Center, San Diego State
00024 University. All rights reserved.
00025
00026 Redistribution and use in source and binary forms, with or without modification,
00027 are permitted provided that the following conditions are met:
00028
00029 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00030 and a copy of the modified files should be reported once modifications are
00031 completed, unless these modifications are made through the project's GitHub
00032 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00033 should be developed and included in any deliverable.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions in binary form must reproduce the above copyright notice,
00039 this list of conditions and the following disclaimer in the documentation and/or
00040 other materials provided with the distribution.
00041
00042 4. Usage of the binary form on proprietary applications shall require explicit
00043 prior written permission from the the copyright holders, and due credit should
00044 be given to the copyright holders.
00045
00046 5. Neither the name of the copyright holder nor the names of its contributors
00047 may be used to endorse or promote products derived from this software without
00048 specific prior written permission.
00049
00050 The copyright holders provide no reassurances that the source code provided does
00051 not infringe any patent, copyright, or any other intellectual property rights of
00052 third parties. The copyright holders disclaim any liability to any recipient for
00053 claims brought against recipient by any third party for infringement of that
00054 parties intellectual property rights.
00055
00056 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00057 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00058 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00059 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00060 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00061 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00062 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00063 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

```

```
00064 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00065 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00066 */
00067
00068 #include <cstring>
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <algorithm>
00074
00075 #include "mtk_foundations.h"
00076 #include "mtk_tools.h"
00077 #include "mtk_dense_matrix.h"
00078 #include "mtk_lapack_adapter.h"
00079
00080 namespace mtk {
00081
00082 extern "C" {
00083
00084 #ifdef MTK_PRECISION_DOUBLE
00085
00104 void dgesv_(int* n,
00105             int* nrhs,
00106             Real* a,
00107             int* lda,
00108             int* ipiv,
00109             Real* b,
00110             int* ldb,
00111             int* info);
00112 #else
00113
00132 void sgesv_(int* n,
00133             int* nrhs,
00134             Real* a,
00135             int* lda,
00136             int* ipiv,
00137             Real* b,
00138             int* ldb,
00139             int* info);
00140 #endif
00141
00142 #ifdef MTK_PRECISION_DOUBLE
00143
00186 void dgels_(char* trans,
00187               int* m,
00188               int* n,
00189               int* nrhs,
00190               Real* a,
00191               int* lda,
00192               Real* b,
00193               int* ldb,
00194               Real* work,
00195               int* lwork,
00196               int* info);
00197 #else
00198
00241 void sgels_(char* trans,
00242               int* m,
00243               int* n,
00244               int* nrhs,
00245               Real* a,
00246               int* lda,
00247               Real* b,
00248               int* ldb,
00249               Real* work,
00250               int* lwork,
00251               int* info);
00252 #endif
00253
00254 #ifdef MTK_PRECISION_DOUBLE
00255
00284 void dgeqrf_(int *m,
00285                 int *n,
00286                 Real *a,
00287                 int *lda,
00288                 Real *tau,
00289                 Real *work,
00290                 int *lwork,
00291                 int *info);
00292 #else
```

```

00293
00322 void sgeqrf_(int *m,
00323             int *n,
00324             Real *a,
00325             int *lda,
00326             Real *tau,
00327             Real *work,
00328             int *lwork,
00329             int *info);
00330 #endif
00331
00332 #ifdef MTK_PRECISION_DOUBLE
00333
00367 void dormqr_(char *side,
00368                 char *trans,
00369                 int *m,
00370                 int *n,
00371                 int *k,
00372                 Real *a,
00373                 int *lda,
00374                 Real *tau,
00375                 Real *c,
00376                 int *ldc,
00377                 Real *work,
00378                 int *lwork,
00379                 int *info);
00380 #else
00381
00415 void sormqr_(char *side,
00416                 char *trans,
00417                 int *m,
00418                 int *n,
00419                 int *k,
00420                 Real *a,
00421                 int *lda,
00422                 Real *tau,
00423                 Real *c,
00424                 int *ldc,
00425                 Real *work,
00426                 int *lwork,
00427                 int *info);
00428 #endif
00429 }
00430 }
00431
00432 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
    mtk::Real *rhs) {
00433
00434
00435 #ifdef MTK_PERFORM_PREVENTIONS
00436 mtk::Tools::Prevent(rhs == nullptr, __FILE__, __LINE__, __func__);
00437 #endif
00438
00439 int *ipiv{}; // Array for pivoting information.
00440 int nrhs{1}; // Number of right-hand sides.
00441 int info{}; // Status of the solution.
00442 int mm_rank{mm.num_rows()}; // Rank of the matrix.
00443
00444 try {
00445     ipiv = new int[mm_rank];
00446 } catch (std::bad_alloc &memory_allocation_exception) {
00447     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00448     std::endl;
00449     std::cerr << memory_allocation_exception.what() << std::endl;
00450 }
00451 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00452
00453 int ldbb = mm_rank;
00454 int mm_ld = mm_rank;
00455
00456 #ifdef MTK_PRECISION_DOUBLE
00457 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00458 #else
00459 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00460 #endif
00461
00462 delete [] ipiv;
00463
00464 return info;
00465 }
00466

```

```

00467 int mtk::LAPACKAdapter::SolveDenseSystem(
00468     mtk::DenseMatrix &mm,
00469                                         mtk::DenseMatrix &bb) {
00470     int nrhs(bb.num_rows()); // Number of right-hand sides.
00471
00472 #ifdef MTK_PERFORM_PREVENTIONS
00473 mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00474 #endif
00475
00476     int *ipiv{}; // Array for pivoting information.
00477     int info{}; // Status of the solution.
00478     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00479
00480     try {
00481         ipiv = new int[mm_rank];
00482     } catch (std::bad_alloc &memory_allocation_exception) {
00483         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00484         std::endl;
00485         std::cerr << memory_allocation_exception.what() << std::endl;
00486     }
00487     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00488
00489     int ldbb = mm_rank;
00490     int mm_ld = mm_rank;
00491
00492 #ifdef MTK_PRECISION_DOUBLE
00493     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00494 #else
00495     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00496 #endif
00497
00498     delete [] ipiv;
00499
00500 // After output, the data in the matrix will be column-major ordered.
00501
00502     bb.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00503
00504 #if MTK_VERBOSE_LEVEL > 12
00505     std::cout << "bb_col_maj_ord =" << std::endl;
00506     std::cout << bb << std::endl;
00507 #endif
00508
00509     bb.OrderRowMajor();
00510
00511 #if MTK_VERBOSE_LEVEL > 12
00512     std::cout << "bb_row_maj_ord =" << std::endl;
00513     std::cout << bb << std::endl;
00514 #endif
00515
00516     return info;
00517 }
00518
00519 int mtk::LAPACKAdapter::SolveDenseSystem(
00520     mtk::DenseMatrix &mm,
00521                                         mtk::UniStgGrid1D &rhs) {
00522     int nrhs{1}; // Number of right-hand sides.
00523
00524     int *ipiv{}; // Array for pivoting information.
00525     int info{}; // Status of the solution.
00526     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00527
00528     try {
00529         ipiv = new int[mm_rank];
00530     } catch (std::bad_alloc &memory_allocation_exception) {
00531         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00532         std::endl;
00533         std::cerr << memory_allocation_exception.what() << std::endl;
00534     }
00535     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00536
00537     int ldbb = mm_rank;
00538     int mm_ld = mm_rank;
00539
00540     mm.OrderColMajor();
00541
00542 #ifdef MTK_PRECISION_DOUBLE
00543     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00544             rhs.discrete_field(), &ldbb, &info);
00545 #else

```

```

00546     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00547             rhs.discrete_field(), &ldbb, &info);
00548 #endif
00549
00550     mm.OrderRowMajor();
00551
00552     delete [] ipiv;
00553
00554     return info;
00555 }
00556
00557 int mtk::LAPACKAdapter::SolveDenseSystem(
00558     mtk::DenseMatrix &mm,
00559                                         mtk::UniStgGrid2D &rhs) {
00560
00561     int nrhs{1}; // Number of right-hand sides.
00562
00563     int *ipiv{}; // Array for pivoting information.
00564     int info{}; // Status of the solution.
00565     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00566
00567     try {
00568         ipiv = new int[mm_rank];
00569     } catch (std::bad_alloc &memory_allocation_exception) {
00570         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00571             std::endl;
00572         std::cerr << memory_allocation_exception.what() << std::endl;
00573     }
00574     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00575
00576     int lddb = mm_rank;
00577     int mm_ld = mm_rank;
00578
00579     mm.OrderColMajor();
00580
00581 #ifdef MTK_PRECISION_DOUBLE
00582     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00583             rhs.discrete_field(), &ldbb, &info);
00584 #else
00585     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00586             rhs.discrete_field(), &ldbb, &info);
00587 #endif
00588
00589     mm.OrderRowMajor();
00590
00591     delete [] ipiv;
00592
00593 }
00594
00595 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
00596     (mtk::DenseMatrix &aa) {
00597
00598     mtk::Real *work{}; // Working array.
00599     mtk::Real *tau{}; // Array for the Householder scalars.
00600
00601     // Prepare to factorize: allocate and inquire for the value of lwork.
00602     try {
00603         work = new mtk::Real[1];
00604     } catch (std::bad_alloc &memory_allocation_exception) {
00605         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00606             std::endl;
00607         std::cerr << memory_allocation_exception.what() << std::endl;
00608     }
00609     memset(work, mtk::kZero, sizeof(aa.data()[0])*1);
00610
00611     int lwork{-1};
00612     int info{};
00613
00614     int aa_num_cols = aa.num_cols();
00615     int aaT_num_rows = aa.num_cols();
00616     int aaT_num_cols = aa.num_rows();
00617
00618 #if MTK_VERBOSE_LEVEL > 12
00619     std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00620     std::cout << aa << std::endl;
00621 #endif
00622
00623 #ifdef MTK_PRECISION_DOUBLE
00624     dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00625             tau,

```

```

00625     work, &lwork, &info);
00626 #else
00627 fgeqrfa_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00628     tau,
00629     work, &lwork, &info);
00630 #endif
00631
00632 if (info == 0) {
00633     lwork = (int) work[0];
00634 } else {
00635     std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00636     std::endl;
00637     std::cerr << "Exiting..." << std::endl;
00638 }
00639
00640 #if MTK_VERBOSE_LEVEL > 10
00641 std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00642     << std::endl;
00643 #endif
00644
00645 delete [] work;
00646 work = nullptr;
00647
00648 // Once we know lwork, we can actually invoke the factorization:
00649 try {
00650     work = new mtk::Real [lwork];
00651 } catch (std::bad_alloc &memory_allocation_exception) {
00652     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00653     std::endl;
00654     std::cerr << memory_allocation_exception.what() << std::endl;
00655 }
00656 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00657
00658 int ltau = std::min(aaT_num_rows,aaT_num_cols);
00659
00660 try {
00661     tau = new mtk::Real [ltau];
00662 } catch (std::bad_alloc &memory_allocation_exception) {
00663     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00664     std::endl;
00665     std::cerr << memory_allocation_exception.what() << std::endl;
00666 }
00667 memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00668
00669 #ifdef MTK_PRECISION_DOUBLE
00670 dgeqrfa_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00671     tau, work, &lwork, &info);
00672 #else
00673 fgeqrfa_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00674     tau, work, &lwork, &info);
00675 #endif
00676
00677 #ifdef MTK_PERFORM_PREVENTIONS
00678 if (!info) {
00679     std::cout << "QR factorization completed!" << std::endl << std::endl;
00680 } else {
00681     std::cerr << "Error solving system! info = " << info << std::endl;
00682     std::cerr << "Exiting..." << std::endl;
00683 }
00684 #endif
00685
00686 #if MTK_VERBOSE_LEVEL > 12
00687 std::cout << "Input matrix AFTER QR factorization:" << std::endl;
00688 std::cout << aa << std::endl;
00689 #endif
00690
00691 // We now generate the real matrix Q with orthonormal columns. This has to
00692 // be done separately since the actual output of dgeqrfa_ (AA_) represents
00693 // the orthogonal matrix Q as a product of min(aa_num_rows, aa_num_cols)
00694 // elementary Householder reflectors. Notice that we must re-inquire the new
00695 // value for lwork that is used.
00696
00697 bool padded{false};
00698
00699 bool transpose{false};
00700
00701 mtk::DenseMatrix QQ_(aa.num_cols(),padded,transpose);
00702
00703 #if MTK_VERBOSE_LEVEL > 12
00704 std::cout << "Initialized QQ_T: " << std::endl;
00705 std::cout << QQ_ << std::endl;

```

```

00706  #endif
00707
00708 // Assemble the QQ_ matrix:
00709 lwork = -1;
00710
00711 delete[] work;
00712 work = nullptr;
00713
00714 try {
00715     work = new mtk::Real[1];
00716 } catch (std::bad_alloc &memory_allocation_exception) {
00717     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00718     std::endl;
00719     std::cerr << memory_allocation_exception.what() <<
00720     std::endl;
00721 }
00722 memset(work, mtk::kZero, sizeof(work[0])*1);
00723
00724 char side_{'L'};
00725 char trans_{'N'};
00726
00727 int aux = QQ_.num_rows();
00728
00729 #ifdef MTK_PRECISION_DOUBLE
00730 dormqr_(&side_, &trans_,
00731         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00732         QQ_.data(), &aux, work, &lwork, &info);
00733 #else
00734 formqr_(&side_, &trans_,
00735         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00736         QQ_.data(), &aux, work, &lwork, &info);
00737#endif
00738
00739 if (info == 0) {
00740     lwork = (int) work[0];
00741 } else {
00742     std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00743     std::cerr << "Exiting..." << std::endl;
00744 }
00745
00746 #if MTK_VERBOSE_LEVEL > 10
00747 std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00748     std::endl << std::endl;
00749#endif
00750
00751 delete[] work;
00752 work = nullptr;
00753
00754 try {
00755     work = new mtk::Real[lwork];
00756 } catch (std::bad_alloc &memory_allocation_exception) {
00757     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00758     std::endl;
00759     std::cerr << memory_allocation_exception.what() << std::endl;
00760 }
00761 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00762
00763 #ifdef MTK_PRECISION_DOUBLE
00764 dormqr_(&side_, &trans_,
00765         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00766         QQ_.data(), &aux, work, &lwork, &info);
00767 #else
00768 formqr_(&side_, &trans_,
00769         &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00770         QQ_.data(), &aux, work, &lwork, &info);
00771#endif
00772
00773 #ifdef MTK_PERFORM_PREVENTIONS
00774 if (!info) {
00775     std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00776 } else {
00777     std::cerr << "Something went wrong solving system! info = " << info <<
00778     std::endl;
00779     std::cerr << "Exiting..." << std::endl;
00780 }
00781#endif
00782
00783 delete[] work;
00784 work = nullptr;
00785
00786 delete[] tau;

```

```

00787     tau = nullptr;
00788
00789     return QQ_;
00790 }
00791
00792 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
00793     mtk::DenseMatrix &aa,
00794                                         mtk::Real *ob_,
00795                                         int ob_ld_) {
00796
00797     // We first invoke the solver to query for the value of lwork. For this,
00798     // we must at least allocate enough space to allow access to WORK(1), or
00799     // work[0]:
00800
00801     // If LWORK = -1, then a workspace query is assumed; the routine only
00802     // calculates the optimal size of the WORK array, returns this value as
00803     // the first entry of the WORK array, and no error message related to
00804     // LWORK is issued by XERBLA.
00805
00806     mtk::Real *work{}; // Work array.
00807
00808     try {
00809         work = new mtk::Real[1];
00810     } catch (std::bad_alloc &memory_allocation_exception) {
00811         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00812             std::endl;
00813         std::cerr << memory_allocation_exception.what() << std::endl;
00814     }
00815     memset(work, mtk::kZero, sizeof(work[0])*1);
00816
00817     char trans_{'N'};
00818     int nrhs_{1};
00819     int info{0};
00820     int lwork{-1};
00821
00822     int AA_num_rows_ = aa.num_cols();
00823     int AA_num_cols_ = aa.num_rows();
00824     int AA_ld_ = std::max(1,aa.num_cols());
00825
00826 #ifdef MTK_PRECISION_DOUBLE
00827     dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00828            ob_, &ob_ld_,
00829            work, &lwork, &info);
00830 #else
00831     sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00832            ob_, &ob_ld_,
00833            work, &lwork, &info);
00834 #endif
00835
00836     if (info == 0) {
00837         lwork = (int) work[0];
00838     } else {
00839         std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00840             std::endl;
00841         std::cerr << "Exiting..." << std::endl;
00842         return info;
00843     }
00844
00845 #if MTK_VERBOSE_LEVEL > 10
00846     std::cout << "lwork = " << std::endl << std::setw(12)<< lwork <<
00847             std::endl << std::endl;
00848 #endif
00849
00850     // We then use lwork's new value to create the work array:
00851     delete[] work;
00852     work = nullptr;
00853
00854     try {
00855         work = new mtk::Real[lwork];
00856     } catch (std::bad_alloc &memory_allocation_exception) {
00857         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00858         std::cerr << memory_allocation_exception.what() << std::endl;
00859     }
00860     memset(work, 0.0, sizeof(work[0])*lwork);
00861
00862     // We now invoke the solver again:
00863 #ifdef MTK_PRECISION_DOUBLE
00864     dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00865            ob_, &ob_ld_,
00866            work, &lwork, &info);
00867 #else

```

```

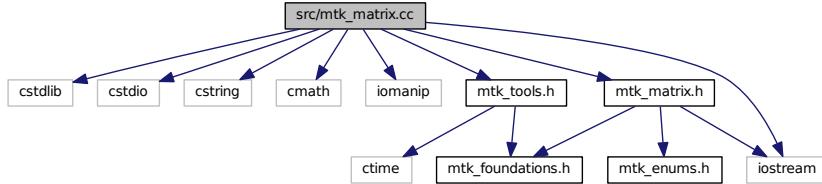
00867     sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00868             ob_, &ob_ld_,
00869             work, &lwork, &info);
00870 #endif
00871
00872 delete [] work;
00873 work = nullptr;
00874
00875 return info;
00876 }
```

18.119 src/mtk_matrix.cc File Reference

Definition of the representation of a matrix in the MTK.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <iostream>
#include "mtk_tools.h"
#include "mtk_matrix.h"
```

Include dependency graph for mtk_matrix.cc:



18.119.1 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.cc](#).

18.120 mtk_matrix.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016 
```

```

00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csdc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <cstdlib>
00057 #include <cstdio>
00058 #include <cstring>
00059 #include <cmath>
00060
00061 #include <iomanip>
00062 #include <iostream>
00063
00064 #include "mtk_tools.h"
00065 #include "mtk_matrix.h"
00066
00067 mtk::Matrix::Matrix():
00068     encoded_operator_(mtk::EncodedOperator::NOOP),
00069     storage_(mtk::MatrixStorage::DENSE),
00070     ordering_(mtk::MatrixOrdering::ROW_MAJOR),
00071     num_rows_(),
00072     num_cols_(),
00073     num_values_(),
00074     leading_dimension_(),
00075     num_low_diags_(),
00076     num_upp_diags_(),
00077     bandwidth_(),
00078     num_null_(),
00079     num_non_null_(),
00080     abs_density_(mtk::kZero),
00081     abs_sparsity_(mtk::kZero),
00082     num_zero_(),
00083     num_non_zero_(),
00084     rel_density_(mtk::kZero),
00085     rel_sparsity_(mtk::kZero) {}
00086
00087 mtk::Matrix::Matrix(const Matrix &in):
00088     encoded_operator_(in.encoded_operator_),
00089     storage_(in.storage_),
00090     ordering_(in.ordering_),
00091     num_rows_(in.num_rows_),
00092     num_cols_(in.num_cols_),
00093     num_values_(in.num_values_),
00094     leading_dimension_(in.leading_dimension_),
00095     num_low_diags_(in.num_low_diags_),
00096     num_upp_diags_(in.num_upp_diags_),
00097     bandwidth_(in.bandwidth_),
00098
00099

```

```
00098     num_null_(in.num_null_),
00099     num_non_null_(in.num_non_null_),
00100     abs_density_(in.abs_density_),
00101     abs_sparsity_(in.abs_sparsity_),
00102     num_zero_(in.num_zero_),
00103     num_non_zero_(in.num_non_zero_),
00104     rel_density_(in.rel_density_),
00105     rel_sparsity_(in.rel_sparsity_) {}
00106
00107     mtk::Matrix::~Matrix() noexcept {}
00108
00109     mtk::EncodedOperator mtk::Matrix::encoded_operator() const
00110         noexcept {
00111         return encoded_operator_;
00112     }
00113
00114     mtk::MatrixStorage mtk::Matrix::storage() const noexcept {
00115
00116         return storage_;
00117     }
00118
00119     mtk::MatrixOrdering mtk::Matrix::ordering() const noexcept {
00120
00121         return ordering_;
00122     }
00123
00124     int mtk::Matrix::num_rows() const noexcept {
00125
00126         return num_rows_;
00127     }
00128
00129     int mtk::Matrix::num_cols() const noexcept {
00130
00131         return num_cols_;
00132     }
00133
00134     int mtk::Matrix::num_values() const noexcept {
00135
00136         return num_values_;
00137     }
00138
00139     int mtk::Matrix::leading_dimension() const noexcept {
00140
00141         return leading_dimension_;
00142     }
00143
00144     int mtk::Matrix::num_zero() const noexcept {
00145
00146         return num_zero_;
00147     }
00148
00149     int mtk::Matrix::num_non_zero() const noexcept {
00150
00151         return num_non_zero_;
00152     }
00153
00154     int mtk::Matrix::num_null() const noexcept {
00155
00156         return num_null_;
00157     }
00158
00159     int mtk::Matrix::num_non_null() const noexcept {
00160
00161         return num_non_null_;
00162     }
00163
00164     int mtk::Matrix::num_low_diags() const noexcept {
00165
00166         return num_low_diags_;
00167     }
00168
00169     int mtk::Matrix::num_upp_diags() const noexcept {
00170
00171         return num_upp_diags_;
00172     }
00173
00174     int mtk::Matrix::bandwidth() const noexcept {
00175
00176         return bandwidth_;
00177     }
```

```

00178
00179 mtk::Real mtk::Matrix::abs_density() const noexcept {
00180
00181     return abs_density_;
00182 }
00183
00184 mtk::Real mtk::Matrix::rel_density() const noexcept {
00185
00186     return rel_density_;
00187 }
00188
00189 mtk::Real mtk::Matrix::abs_sparsity() const noexcept {
00190
00191     return abs_sparsity_;
00192 }
00193
00194 mtk::Real mtk::Matrix::rel_sparsity() const noexcept {
00195
00196     return rel_sparsity_;
00197 }
00198
00199 void mtk::Matrix::set_encoded_operator(const
00200     mtk::EncodedOperator &in)
00201     noexcept {
00202
00203 #ifdef MTK_PERFORM_PREVENTIONS
00204     bool aux = (in != mtk::EncodedOperator::NOOP) &&
00205         (in != mtk::EncodedOperator::GRADIENT) &&
00206         (in != mtk::EncodedOperator::DIVERGENCE) &&
00207         (in != mtk::EncodedOperator::INTERPOLATION) &&
00208         (in != mtk::EncodedOperator::CURL) &&
00209         (in != mtk::EncodedOperator::LAPLACIAN);
00210
00211     mtk::Tools::Prevent(aux, __FILE__, __LINE__, __func__);
00212 #endif
00213
00214     encoded_operator_ = in;
00215 }
00216
00217 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss)
00218     noexcept {
00219
00220 #ifdef MTK_PERFORM_PREVENTIONS
00221     mtk::Tools::Prevent(!!(ss == mtk::MatrixStorage::DENSE ||
00222                         ss == mtk::MatrixStorage::BANDED ||
00223                         ss == mtk::MatrixStorage::CRS),
00224                         __FILE__, __LINE__, __func__);
00225
00226     storage_ = ss;
00227 }
00228
00229 void mtk::Matrix::set_ordering(const
00230     mtk::MatrixOrdering &oo) noexcept {
00231
00232 #ifdef MTK_PERFORM_PREVENTIONS
00233     bool aux{oo == mtk::MatrixOrdering::ROW_MAJOR ||
00234             oo == mtk::MatrixOrdering::COL_MAJOR};
00235     mtk::Tools::Prevent(!aux, __FILE__, __LINE__, __func__);
00236
00237     ordering_ = oo;
00238
00239     leading_dimension_ = ComputeLeadingDimension(num_rows_, num_cols_);
00240 }
00241
00242 void mtk::Matrix::set_num_rows(const int &num_rows) noexcept {
00243
00244 #ifdef MTK_PERFORM_PREVENTIONS
00245     mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00246
00247     num_rows_ = num_rows;
00248     num_values_ = ComputeNumValues(num_rows_, num_cols_);
00249     leading_dimension_ = ComputeLeadingDimension(num_rows_, num_cols_);
00250
00251     num_null_ = num_values_;
00252     num_non_null_ = 0;
00253     abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00254     abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00255 }
```

```

00256     num_zero_ = 0;
00257     num_non_zero_ = num_values_;
00258     rel_density_ = ComputeRelDensity(num_non_zero_, num_values_);
00259     rel_sparsity_ = ComputeRelSparsity(rel_density_);
00260 }
00261
00262 void mtk::Matrix::set_num_cols(const int &num_cols) noexcept {
00263
00264 #ifdef MTK_PERFORM_PREVENTIONS
00265     mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00266 #endif
00267
00268     num_cols_ = num_cols;
00269     num_values_ = ComputeNumValues(num_rows_, num_cols_);
00270     leading_dimension_ = ComputeLeadingDimension(num_rows_, num_cols_);
00271
00272     num_null_ = num_values_;
00273     num_non_null_ = 0;
00274     abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00275     abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00276
00277     num_zero_ = 0;
00278     num_non_zero_ = num_values_;
00279     rel_density_ = ComputeRelDensity(num_non_zero_, num_values_);
00280     rel_sparsity_ = ComputeRelSparsity(rel_density_);
00281 }
00282
00283 void mtk::Matrix::set_num_low_diags(const int &num_low_diags) noexcept {
00284
00285 #ifdef MTK_PERFORM_PREVENTIONS
00286     mtk::Tools::Prevent(num_low_diags < 0, __FILE__, __LINE__, __func__);
00287 #endif
00288
00289     num_low_diags_ = num_low_diags;
00290     bandwidth_ = ComputeBandwidth(num_low_diags_, num_upp_diags_);
00291 }
00292
00293 void mtk::Matrix::set_num_upp_diags(const int &num_upp_diags) noexcept {
00294
00295 #ifdef MTK_PERFORM_PREVENTIONS
00296     mtk::Tools::Prevent(num_upp_diags < 0, __FILE__, __LINE__, __func__);
00297 #endif
00298
00299     num_upp_diags_ = num_upp_diags;
00300     bandwidth_ = ComputeBandwidth(num_low_diags_, num_upp_diags_);
00301 }
00302
00303 void mtk::Matrix::set_num_null(const int &num_null) noexcept {
00304
00305 #ifdef MTK_PERFORM_PREVENTIONS
00306     mtk::Tools::Prevent(num_null < 0, __FILE__, __LINE__, __func__);
00307 #endif
00308
00309     num_null_ = num_null;
00310     num_non_null_ = ComputeNumNonNull(num_values_, num_null_);
00311
00312     abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00313     abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00314 }
00315
00316 void mtk::Matrix::set_num_zero(const int &num_zero) noexcept {
00317
00318 #ifdef MTK_PERFORM_PREVENTIONS
00319     mtk::Tools::Prevent(num_zero < 0, __FILE__, __LINE__, __func__);
00320 #endif
00321
00322     num_zero_ = num_zero;
00323     num_non_zero_ = ComputeNumNonZero(num_values_, num_zero_);
00324     rel_density_ = ComputeRelDensity(num_non_zero_, num_values_);
00325     rel_sparsity_ = ComputeRelSparsity(rel_density_);
00326
00327     num_null_ = num_null_ - num_zero_;
00328     num_non_null_ = ComputeNumNonNull(num_values_, num_null_);
00329     abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00330     abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00331 }
00332
00333 void mtk::Matrix::IncreaseNumNull() noexcept {
00334
00335 #ifdef MTK_PERFORM_PREVENTIONS
00336     mtk::Tools::Prevent(num_null_ == num_values_, __FILE__, __LINE__, __func__);
00337 
```

```

00337 #endif
00338 num_null_++;
00339 num_non_null_ = ComputeNumNonNull(num_values_, num_null_);
00340 abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00341 abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00342 }
00343
00344 }
00345
00346 void mtk::Matrix::DecreaseNumNull() noexcept {
00347 #ifdef MTK_PERFORM_PREVENTIONS
00348 mtk::Tools::Prevent(num_null_ == 0, __FILE__, __LINE__, __func__);
00349 #endif
00350
00351 num_null--;
00352 num_non_null_ = ComputeNumNonNull(num_values_, num_null_);
00353 abs_density_ = ComputeAbsDensity(num_non_null_, num_values_);
00354 abs_sparsity_ = ComputeAbsSparsity(abs_density_);
00355 }
00356
00357 }
00358
00359 void mtk::Matrix::IncreaseNumZero() noexcept {
00360 #ifdef MTK_PERFORM_PREVENTIONS
00361 mtk::Tools::Prevent(num_zero_ == num_values_, __FILE__, __LINE__, __func__);
00362 #endif
00363
00364 num_zero++;
00365 num_non_zero_ = ComputeNumNonZero(num_values_, num_zero_);
00366 rel_density_ = ComputeRelDensity(num_non_zero_, num_values_);
00367 rel_sparsity_ = ComputeRelSparsity(rel_density_);
00368
00369 DecreaseNumNull();
00370
00371 }
00372 }
00373
00374 void mtk::Matrix::DecreaseNumZero() noexcept {
00375 #ifdef MTK_PERFORM_PREVENTIONS
00376 mtk::Tools::Prevent(num_zero_ == 0, __FILE__, __LINE__, __func__);
00377 #endif
00378
00379 num_zero--;
00380 num_non_zero_ = ComputeNumNonZero(num_values_, num_zero_);
00381 rel_density_ = ComputeRelDensity(num_non_zero_, num_values_);
00382 rel_sparsity_ = ComputeRelSparsity(rel_density_);
00383
00384 IncreaseNumNull();
00385
00386 }
00387 }
00388
00389 int mtk::Matrix::ComputeNumValues(const int &num_rows,
00390                                     const int &num_cols) const noexcept {
00391 #ifdef MTK_PERFORM_PREVENTIONS
00392 mtk::Tools::Prevent(num_rows < 0, __FILE__, __LINE__, __func__);
00393 mtk::Tools::Prevent(num_cols < 0, __FILE__, __LINE__, __func__);
00394 #endif
00395
00396 return num_rows*num_cols;
00397 }
00398 }
00399
00400 int mtk::Matrix::ComputeLeadingDimension(const int &num_rows,
00401                                         const int &num_cols) const noexcept {
00402
00403 #ifdef MTK_PERFORM_PREVENTIONS
00404 mtk::Tools::Prevent(num_rows < 0, __FILE__, __LINE__, __func__);
00405 mtk::Tools::Prevent(num_cols < 0, __FILE__, __LINE__, __func__);
00406 #endif
00407
00408 return (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00409         std::max(1,num_cols): std::max(1,num_rows);
00410 }
00411
00412 int mtk::Matrix::ComputeBandwidth(const int &num_low_diags,
00413                                   const int &num_upp_diags) const noexcept {
00414
00415 #ifdef MTK_PERFORM_PREVENTIONS
00416 mtk::Tools::Prevent(num_low_diags < 0, __FILE__, __LINE__, __func__);
00417 mtk::Tools::Prevent(num_upp_diags < 0, __FILE__, __LINE__, __func__);
00418

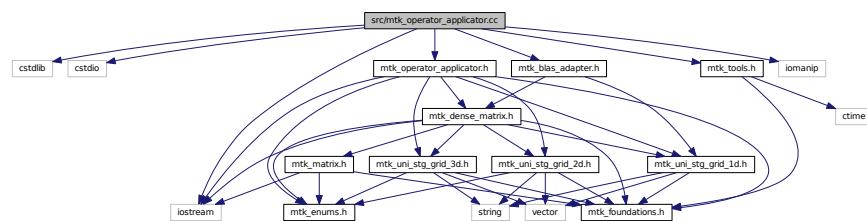
```

```
00418     #endif
00419     return num_low_diags + 1 + num_upp_diags;
00420 }
00422
00423 int mtk::Matrix::ComputeNumNonNull(const int &num_values, const int &num_null
00424 )
00425     const noexcept {
00426
00427     #ifdef MTK_PERFORM_PREVENTIONS
00428         mtk::Tools::Prevent(num_values < 0, __FILE__, __LINE__, __func__);
00429         mtk::Tools::Prevent(num_null < 0, __FILE__, __LINE__, __func__);
00430     #endif
00431
00432     return num_values - num_null;
00433 }
00434
00435 mtk::Real mtk::Matrix::ComputeAbsDensity(const int &num_non_null,
00436                                         const int &num_values) const noexcept {
00437
00438     #ifdef MTK_PERFORM_PREVENTIONS
00439         mtk::Tools::Prevent(num_non_null < 0, __FILE__, __LINE__, __func__);
00440         mtk::Tools::Prevent(num_values < 0, __FILE__, __LINE__, __func__);
00441     #endif
00442
00443     return (num_values == mtk::kZero)?
00444         mtk::kZero: (mtk::Real) num_non_null/num_values;
00445 }
00446
00447 mtk::Real mtk::Matrix::ComputeAbsSparsity(const
00448     mtk::Real &absolute_density)
00449     const noexcept {
00450
00451     #ifdef MTK_PERFORM_PREVENTIONS
00452         mtk::Tools::Prevent(absolute_density < 0, __FILE__, __LINE__, __func__);
00453     #endif
00454
00455     return mtk::kOne - absolute_density;
00456 }
00457
00458 int mtk::Matrix::ComputeNumNonZero(const int &num_values, const int &num_zero
00459 )
00460     const noexcept {
00461
00462     #ifdef MTK_PERFORM_PREVENTIONS
00463         mtk::Tools::Prevent(num_values < 0, __FILE__, __LINE__, __func__);
00464         mtk::Tools::Prevent(num_zero < 0, __FILE__, __LINE__, __func__);
00465     #endif
00466
00467     return num_values - num_zero;
00468 }
00469
00470 mtk::Real mtk::Matrix::ComputeRelDensity(const int &num_non_zero,
00471                                         const int &num_values) const noexcept {
00472
00473     #ifdef MTK_PERFORM_PREVENTIONS
00474         mtk::Tools::Prevent(num_non_zero < 0, __FILE__, __LINE__, __func__);
00475         mtk::Tools::Prevent(num_values < 0, __FILE__, __LINE__, __func__);
00476     #endif
00477
00478     return (num_values == mtk::kZero)?
00479         mtk::kZero: mtk::kOne - (mtk::Real) num_non_zero/num_values;
00480 }
00481
00482 mtk::Real mtk::Matrix::ComputeRelSparsity(const
00483     mtk::Real &relative_density)
00484     const noexcept {
00485
00486     #ifdef MTK_PERFORM_PREVENTIONS
00487         mtk::Tools::Prevent(relative_density < 0, __FILE__, __LINE__, __func__);
00488     #endif
00489
00490     return mtk::kOne - relative_density;
00491 }
```

18.121 src/mtk_operator_applicator.cc File Reference

Implements the class OperatorApplicator.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_tools.h"
#include "mtk blas_adapter.h"
#include "mtk_operator_applicator.h"
Include dependency graph for mtk_operator_applicator.cc:
```



18.121.1 Detailed Description

Author

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

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

18.122 mtk_operator_applicator.cc

```

00001
00002  /*
00003 Copyright (C) 2016, Computational Science Research Center, San Diego State
00004 University. All rights reserved.
00005
00006 Redistribution and use in source and binary forms, with or without modification,
00007 are permitted provided that the following conditions are met:
00008
00009 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00010 and a copy of the modified files should be reported once modifications are
00011 completed, unless these modifications are made through the project's GitHub
00012 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00013 should be developed and included in any deliverable.
00014
00015 2. Redistributions of source code must be done through direct
00016 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00017
00018 3. Redistributions in binary form must reproduce the above copyright notice,
00019 this list of conditions and the following disclaimer in the documentation and/or
00020 other materials provided with the distribution.
00021
00022 4. Usage of the binary form on proprietary applications shall require explicit
00023 prior written permission from the the copyright holders, and due credit should
00024 be given to the copyright holders.
00025
00026 5. Neither the name of the copyright holder nor the names of its contributors
00027 may be used to endorse or promote products derived from this software without
00028 specific prior written permission.
00029
00030
00031
00032
00033
00034
  
```

```

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 #include <cstdlib>
00055 #include <cstdio>
00056
00057 #include <iostream>
00058 #include <iomanip>
00059
00060 #include "mtk_tools.h"
00061 #include "mtk_blas_adapter.h"
00062 #include "mtk_operator_applicator.h"
00063
00064 void mtk::OperatorApplicator::ApplyDenseMatrixGradientOn1DGrid
(
00065     DenseMatrix &grad,
00066     UniStgGrid1D &grid,
00067     UniStgGrid1D &out) {
00068
00069 #if MTK_PERFORM_PREVENTIONS
00070     mtk::Tools::Prevent(grad.encoded_operator() !=
00071                         mtk::EncodedOperator::GRADIENT,
00072                         __FILE__, __LINE__, __func__);
00073     mtk::Tools::Prevent(grid.field_nature() !=
00074                         mtk::FieldNature::SCALAR,
00075                         __FILE__, __LINE__, __func__);
00076     mtk::Tools::Prevent(out.field_nature() !=
00077                         mtk::FieldNature::VECTOR,
00078                         __FILE__, __LINE__, __func__);
00079 #endif
00080
00081     mtk::Real alpha{mtk::kOne};
00082     mtk::Real beta{mtk::kZero};
00083
00084     mtk::BLASAdapter::RealDenseMV(alpha,
00085                                     grad,
00086                                     grid.discrete_field(),
00087                                     beta,
00088                                     out.discrete_field());
00089
00090 void mtk::OperatorApplicator::ApplyDenseMatrixDivergenceOn1DGrid
(
00091     DenseMatrix &div,
00092     UniStgGrid1D &grid,
00093     UniStgGrid1D &out) {
00094
00095 #if MTK_PERFORM_PREVENTIONS
00096     mtk::Tools::Prevent(div.encoded_operator() !=
00097                         mtk::EncodedOperator::DIVERGENCE,
00098                         __FILE__, __LINE__, __func__);
00099     mtk::Tools::Prevent(grid.field_nature() !=
00100                         mtk::FieldNature::VECTOR,
00101                         __FILE__, __LINE__, __func__);
00102
00103     mtk::Real alpha{mtk::kOne};
00104     mtk::Real beta{mtk::kZero};
00105
00106     mtk::BLASAdapter::RealDenseMV(alpha,
00107                                     div,
00108                                     grid.discrete_field()),

```

```

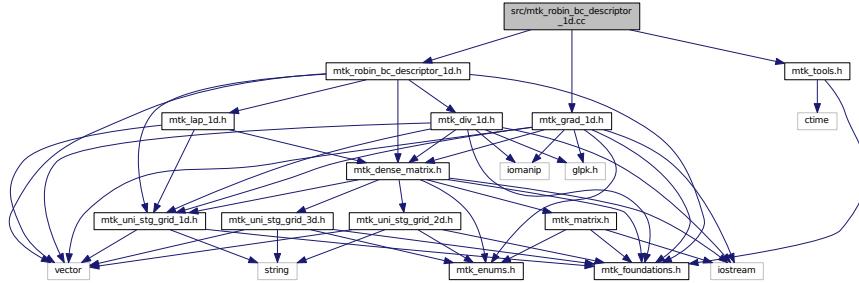
00109             beta,
00110         out.discrete_field());
00111     }
00112
00113 void mtk::OperatorApplicator::ApplyDenseMatrixLaplacianOn1DGrid
(
00114     DenseMatrix &lap,
00115     UniStgGrid1D &grid,
00116     UniStgGrid1D &out) {
00117
00118 #if MTK_PERFORM_PREVENTIONS
00119     mtk::Tools::Prevent(lap.encoded_operator() !=
00120                         __FILE__, __LINE__, __func__);
00121     mtk::Tools::Prevent(grid.field_nature() !=
00122                         mtk::FieldNature::SCALAR,
00123                         __FILE__, __LINE__, __func__);
00124     mtk::Tools::Prevent(out.field_nature() !=
00125                         mtk::FieldNature::SCALAR,
00126                         __FILE__, __LINE__, __func__);
00127 #endif
00128     mtk::Real alpha(mtk::kOne);
00129     mtk::Real beta(mtk::kZero);
00130
00131     mtk::BLASAdapter::RealDenseMV(alpha,
00132                                     lap,
00133                                     grid.discrete_field(),
00134                                     beta,
00135                                     out.discrete_field());

```

18.123 src/mtk_robin_bc_descriptor_1d.cc File Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include "mtk_tools.h"
#include "mtk_grad_1d.h"
#include "mtk_robin_bc_descriptor_1d.h"
Include dependency graph for mtk_robin_bc_descriptor_1d.cc:
```



18.123.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on $\partial\Omega$** if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) = \beta_b(b, t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

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

Definition in file [mtk_robin_bc_descriptor_1d.cc](#).

18.124 mtk_robin_bc_descriptor_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2016, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

```

```

00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include "mtk_tools.h"
00090 #include "mtk_grad_1d.h"
00091 #include "mtk_robin_bc_descriptor_1d.h"
00092
00093 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D():
00094     highest_order_diff_west_{-1},
00095     highest_order_diff_east_{-1},
00096     west_condition_{nullptr},
00097     east_condition_{nullptr} {}
00098
00099 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D(
00100     const mtk::RobinBCDescriptor1D &desc):
00101     highest_order_diff_west_(desc.highest_order_diff_west_),
00102     highest_order_diff_east_(desc.highest_order_diff_east_),
00103     west_condition_(desc.west_condition_),
00104     east_condition_(desc.east_condition_) {}
00105
00106 mtk::RobinBCDescriptor1D::~RobinBCDescriptor1D() noexcept {}
00107
00108 int mtk::RobinBCDescriptor1D::highest_order_diff_west()
00109     const noexcept {
00110     return highest_order_diff_west_;
00111 }
00112
00113 int mtk::RobinBCDescriptor1D::highest_order_diff_east()
00114     const noexcept {
00115     return highest_order_diff_east_;
00116 }
00117
00118 void mtk::RobinBCDescriptor1D::PushBackWestCoeff(
00119     mtk::CoefficientFunction0D cw) {
00120
00121 #ifdef MTK_PERFORM_PREVENTIONS
00122     mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00123     mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00124                         __FILE__, __LINE__, __func__);
00125 #endif
00126
00127     west_coefficients_.push_back(cw);
00128
00129     highest_order_diff_west_++;
00130 }
00131
00132 void mtk::RobinBCDescriptor1D::PushBackEastCoeff(
00133     mtk::CoefficientFunction0D ce) {
00134
00135 #ifdef MTK_PERFORM_PREVENTIONS
00136     mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00137     mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00138                         __FILE__, __LINE__, __func__);
00139 #endif
00140
00141     east_coefficients_.push_back(ce);
00142
00143     highest_order_diff_east_++;
00144 }
00145
00146 void mtk::RobinBCDescriptor1D::set_west_condition(
00147     mtk::Real (*west_condition)(const mtk::Real &tt)) noexcept {
00148
00149 #ifdef MTK_PERFORM_PREVENTIONS
00150     mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00151 #endif
00152
00153     west_condition_ = west_condition;
00154 }
00155
00156 void mtk::RobinBCDescriptor1D::set_east_condition(
00157     mtk::Real (*east_condition)(const mtk::Real &tt)) noexcept {
00158
00159 #ifdef MTK_PERFORM_PREVENTIONS
00160     mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00161 #endif

```

```

00162     east_condition_ = east_condition;
00163 }
00165
00166 bool mtk::RobinBCDescriptor1D::ImposeOnDivergenceMatrix(
00167     const mtk::DivID &div,
00168     mtk::DenseMatrix &matrix,
00169     const std::vector<Real> &parameters,
00170     const mtk::Real &time) const {
00171
00172 #ifdef MTK_PERFORM_PREVENTIONS
00173     mtk::Tools::Prevent(highest_order_diff_west_ == -1,
00174         __FILE__, __LINE__, __func__);
00175     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00176         __FILE__, __LINE__, __func__);
00177     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00178     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00179 #endif
00180
00182
00183     std::vector<mtk::Real> aux_pp(parameters);
00184
00186     matrix.SetValue(0, 0, (west_coefficients_[0])(time, aux_pp));
00187
00189     matrix.SetValue(matrix.num_rows() - 1,
00190         matrix.num_cols() - 1,
00191         (east_coefficients_[0])(time, aux_pp));
00192
00194
00195     if (highest_order_diff_west_ > 0) {}
00196
00197     return true;
00198 }
00199
00200 bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix(
00201     const mtk::Lap1D &lap,
00202     mtk::DenseMatrix &matrix,
00203     const std::vector<Real> &parameters,
00204     const mtk::Real &time) const {
00205
00206 #ifdef MTK_PERFORM_PREVENTIONS
00207     mtk::Tools::Prevent(highest_order_diff_west_ == -1,
00208         __FILE__, __LINE__, __func__);
00209     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00210         __FILE__, __LINE__, __func__);
00211     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00212     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00213 #endif
00214
00216
00217     std::vector<mtk::Real> aux_pp(parameters);
00218
00220     matrix.SetValue(0, 0, (west_coefficients_[0])(time, aux_pp));
00221
00223     matrix.SetValue(matrix.num_rows() - 1,
00224         matrix.num_cols() - 1,
00225         (east_coefficients_[0])(time, aux_pp));
00226
00228
00229     if (highest_order_diff_west_ > 0) {
00230
00232         mtk::Grad1D grad;
00233         if (!grad.ConstructGrad1D(lap.order_accuracy(),
00234             lap.mimetic_threshold())) {
00235             return false;
00236         }
00237
00239
00243         mtk::DenseMatrix coeffs(grad.mim_bndy());
00244
00245         mtk::Real idx = mtk::kOne/lap.delta();
00246
00248         for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00249             mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00250             mtk::Real unit_normal{-mtk::kOne};
00251             aux *= unit_normal*(west_coefficients_[1])(time, aux_pp);
00252             matrix.SetValue(0, ii, matrix.GetValue(0, ii) + aux);
00253         }
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265

```

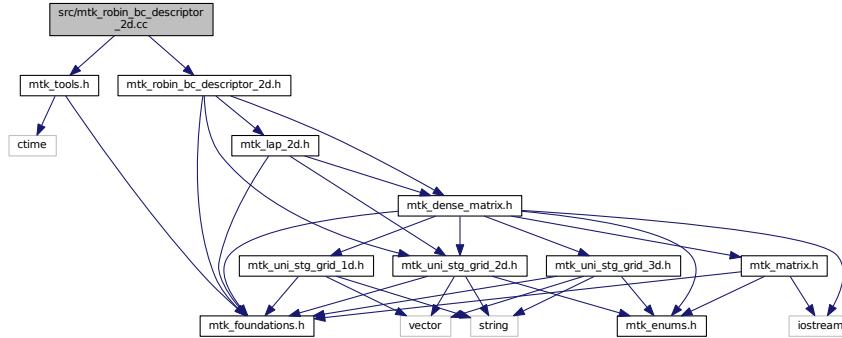
```

00266     for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00267         mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00268         mtk::Real unit_normal{mtk::kOne};
00269         aux *= -unit_normal*(east_coefficients_[1])(time, aux_pp);
00270         matrix.SetValue(matrix.num_rows() - 1,
00271                         matrix.num_rows() - 1 - ii,
00272                         matrix.GetValue(matrix.num_rows() - 1,
00273                                         matrix.num_rows() - 1 - ii) + aux);
00274     }
00275 }
00276 }
00277
00278     return true;
00279 }
00280 }
00281
00282 void mtk::RobinBCDescriptor1D::ImposeOnGrid(
00283     UniStgGrid1D &grid,
00284     const mtk::Real &time) const {
00285
00286 #ifdef MTK_PERFORM_PREVENTIONS
00287     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00288     mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00289     mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00290 #endif
00291     (grid.discrete_field())[0] = west_condition_(time);
00292     (grid.discrete_field())[grid.num_cells_x() + 1] = east_condition_(time);
00293 }
00294
00295 }
```

18.125 src/mtk_robin_bc_descriptor_2d.cc File Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include "mtk_tools.h"
#include "mtk_robin_bc_descriptor_2d.h"
Include dependency graph for mtk_robin_bc_descriptor_2d.cc:
```



18.125.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition on** $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \quad \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

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

Definition in file [mtk_robin_bc_descriptor_2d.cc](#).

18.126 mtk_robin_bc_descriptor_2d.cc

```

00001
00034 

```

```

00083
00084 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D():
00085     highest_order_diff_west_(-1),
00086     highest_order_diff_east_(-1),
00087     highest_order_diff_south_(-1),
00088     highest_order_diff_north_(-1),
00089     west_condition_(),
00090     east_condition_(),
00091     south_condition_(),
00092     north_condition_() {}
00093
00094 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D(
00095     const mtk::RobinBCDescriptor2D &desc):
00096     highest_order_diff_west_(desc.highest_order_diff_west_),
00097     highest_order_diff_east_(desc.highest_order_diff_east_),
00098     highest_order_diff_south_(desc.highest_order_diff_south_),
00099     highest_order_diff_north_(desc.highest_order_diff_north_),
00100    west_condition_(desc.west_condition_),
00101    east_condition_(desc.east_condition_),
00102    south_condition_(desc.south_condition_),
00103    north_condition_(desc.north_condition_) {}
00104
00105 mtk::RobinBCDescriptor2D::~RobinBCDescriptor2D() noexcept {}
00106
00107 int mtk::RobinBCDescriptor2D::highest_order_diff_west()
00108     const noexcept {
00109     return highest_order_diff_west_;
00110 }
00111
00112 int mtk::RobinBCDescriptor2D::highest_order_diff_east()
00113     const noexcept {
00114     return highest_order_diff_east_;
00115 }
00116
00117 int mtk::RobinBCDescriptor2D::highest_order_diff_south()
00118     const noexcept {
00119     return highest_order_diff_south_;
00120 }
00121
00122 int mtk::RobinBCDescriptor2D::highest_order_diff_north()
00123     const noexcept {
00124     return highest_order_diff_north_;
00125 }
00126
00127 void mtk::RobinBCDescriptor2D::PushBackWestCoeff(
00128     mtk::CoefficientFunction1D cw) {
00129
00130 #ifdef MTK_PERFORM_PREVENTIONS
00131     mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00132     mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00133                         __FILE__, __LINE__, __func__);
00134 #endif
00135
00136     west_coefficients_.push_back(cw);
00137
00138     highest_order_diff_west_++;
00139 }
00140
00141 void mtk::RobinBCDescriptor2D::PushBackEastCoeff(
00142     mtk::CoefficientFunction1D ce) {
00143
00144 #ifdef MTK_PERFORM_PREVENTIONS
00145     mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00146     mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00147                         __FILE__, __LINE__, __func__);
00148 #endif
00149
00150     east_coefficients_.push_back(ce);
00151
00152     highest_order_diff_east_++;
00153 }
00154
00155 void mtk::RobinBCDescriptor2D::PushBackSouthCoeff(
00156     mtk::CoefficientFunction1D cs) {
00157
00158 #ifdef MTK_PERFORM_PREVENTIONS
00159     mtk::Tools::Prevent(cs == nullptr, __FILE__, __LINE__, __func__);

```

```

00160     mtk::Tools::Prevent(highest_order_diff_south_ > 1,
00161                           __FILE__, __LINE__, __func__);
00162 #endif
00163
00164     south_coefficients_.push_back(cs);
00165
00166     highest_order_diff_south_++;
00167 }
00168
00169 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff(
00170     mtk::CoefficientFunction1D cn) {
00171
00172 #ifdef MTK_PERFORM_PREVENTIONS
00173     mtk::Tools::Prevent(cn == nullptr, __FILE__, __LINE__, __func__);
00174     mtk::Tools::Prevent(highest_order_diff_north_ > 1,
00175                           __FILE__, __LINE__, __func__);
00176 #endif
00177
00178     north_coefficients_.push_back(cn);
00179
00180     highest_order_diff_north_++;
00181 }
00182
00183 void mtk::RobinBCDescriptor2D::set_west_condition(
00184     mtk::Real (*west_condition)(const mtk::Real &yy,
00185                               const mtk::Real &tt)) noexcept {
00186
00187 #ifdef MTK_PERFORM_PREVENTIONS
00188     mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00189 #endif
00190
00191     west_condition_ = west_condition;
00192 }
00193
00194 void mtk::RobinBCDescriptor2D::set_east_condition(
00195     mtk::Real (*east_condition)(const mtk::Real &yy,
00196                               const mtk::Real &tt)) noexcept {
00197
00198 #ifdef MTK_PERFORM_PREVENTIONS
00199     mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00200 #endif
00201
00202     east_condition_ = east_condition;
00203 }
00204
00205 void mtk::RobinBCDescriptor2D::set_south_condition(
00206     mtk::Real (*south_condition)(const mtk::Real &xx,
00207                               const mtk::Real &tt)) noexcept {
00208
00209 #ifdef MTK_PERFORM_PREVENTIONS
00210     mtk::Tools::Prevent(south_condition == nullptr,
00211                           __FILE__, __LINE__, __func__);
00212 #endif
00213
00214     south_condition_ = south_condition;
00215 }
00216
00217 void mtk::RobinBCDescriptor2D::set_north_condition(
00218     mtk::Real (*north_condition)(const mtk::Real &xx,
00219                               const mtk::Real &tt)) noexcept {
00220
00221 #ifdef MTK_PERFORM_PREVENTIONS
00222     mtk::Tools::Prevent(north_condition == nullptr,
00223                           __FILE__, __LINE__, __func__);
00224 #endif
00225
00226     north_condition_ = north_condition;
00227 }
00228
00229 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace
00230 {
00231     const mtk::Lap2D &lap,
00232     const mtk::UnistgGrid2D &grid,
00233     mtk::DenseMatrix &matrix,
00234     const mtk::Real &time) const {
00235
00236
00237 // For the south-west corner:
00238 auto cc = (south_coefficients_[0])(grid.west_bndy(), time);
00239
00240 #if MTK_VERBOSE_LEVEL > 2

```

```

00241     std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00242         matrix.num_cols() << " columns." << std::endl;
00243     std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00244 #endif
00245
00246     matrix.SetValue(0, 0, cc);
00247
00248 // Compute first centers per dimension.
00249     auto first_center_x = grid.west_bndy() + grid.delta_x() /
00250         mtk::kTwo;
00251
00252 // For each entry on the diagonal (south boundary):
00253     for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00254         // Evaluate next set spatial coordinates to evaluate the coefficient.
00255         mtk::Real xx = first_center_x + ii*grid.delta_x();
00256         // Evaluate and assign the Dirichlet coefficient.
00257         cc = (south_coefficients_[0])(xx, time);
00258
00259 #if MTK_VERBOSE_LEVEL > 2
00260     std::cout << "Setting at " << ii + 1 << ' ' << ii + 1 << std::endl;
00261 #endif
00262
00263     matrix.SetValue(ii + 1, ii + 1, cc);
00264 }
00265
00266 // For the south-east corner:
00267     cc = (south_coefficients_[0])(grid.east_bndy(), time);
00268
00269 #if MTK_VERBOSE_LEVEL > 2
00270     std::cout << "Setting at " << grid.num_cells_x() + 1 << ' ' <<
00271         grid.num_cells_x() + 1 << std::endl;
00272 #endif
00273
00274     matrix.SetValue(grid.num_cells_x() + 1, grid.num_cells_x() + 1,
00275         cc);
00276
00277     if (highest_order_diff_south_ > 0) {
00278 }
00279
00280     return true;
00281 }
00282 }
00283
00284 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace
00285 {
00286     const mtk::Lap2D &lap,
00287     const mtk::UniStgGrid2D &grid,
00288     mtk::DenseMatrix &matrix,
00289     const mtk::Real &time) const {
00290
00291
00292 // For each entry on the diagonal:
00293     for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00294         // Evaluate next set spatial coordinates to evaluate the coefficient.
00295         mtk::Real xx{grid.discrete_domain_x())[ii]};
00296         // Evaluate and assign the Dirichlet coefficient.
00297         mtk::Real cc = (south_coefficients_[0])(xx, time);
00298         matrix.SetValue(ii, ii, cc);
00299     }
00300
00301     if (highest_order_diff_south_ > 0) {
00302 }
00303
00304     return true;
00305 }
00306
00307 }
00308
00309     return true;
00310 }
00311
00312 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace
00313 {
00314     const mtk::Lap2D &lap,
00315     const mtk::UniStgGrid2D &grid,
00316     mtk::DenseMatrix &matrix,
00317     const mtk::Real &time) const {
00318
00319     int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00320
00321
00322 // For the north-west corner:
00323     mtk::Real cc =
00324         (north_coefficients_[0])(grid.west_bndy(), time);

```

```

00325
00326 #if MTK_VERBOSE_LEVEL > 2
00327 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00328 matrix.num_cols() << " columns." << std::endl;
00329 std::cout << "Setting at " << north_offset << ' ' << north_offset <<
00330 std::endl;
00331 #endif
00332
00333 matrix.SetValue(north_offset, north_offset, cc);
00334
00335 // Compute first centers per dimension.
00336 auto first_center_x = grid.west_bndy() + grid.delta_x() /
00337 mtk::kTwo;
00338 // For each entry on the diagonal (north boundary):
00339 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00340 // Evaluate next set spatial coordinates to evaluate the coefficient.
00341 mtk::Real xx = first_center_x + ii*grid.delta_x();
00342 // Evaluate and assign the Dirichlet coefficient.
00343 cc = (north_coefficients_[0])(xx, time);
00344
00345 #if MTK_VERBOSE_LEVEL > 2
00346 std::cout << "Setting at " << north_offset + ii + 1 << ' ' <<
00347 north_offset + ii + 1 << std::endl;
00348 #endif
00349
00350 matrix.SetValue(north_offset + ii + 1, north_offset + ii + 1, cc);
00351 }
00352
00353 // For the north-east corner:
00354 cc = (north_coefficients_[0])(grid.east_bndy(), time);
00355
00356 #if MTK_VERBOSE_LEVEL > 2
00357 std::cout << "Setting at " << north_offset + grid.num_cells_x() + 1 <<
00358 ' ' << north_offset + grid.num_cells_x() + 1 << std::endl;
00359 #endif
00360
00361 matrix.SetValue(north_offset + grid.num_cells_x() + 1,
00362 north_offset + grid.num_cells_x() + 1, cc);
00363
00364 if (highest_order_diff_north_ > 0) {
00365 }
00366
00367 return true;
00368 }
00369
00370 }
00371
00372 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace
00373 (
00374     const mtk::Lap2D &lap,
00375     const mtk::UniStgGrid2D &grid,
00376     mtk::DenseMatrix &matrix,
00377     const mtk::Real &time) const {
00378
00379     int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00380
00381     for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00382         mtk::Real xx{grid.discrete_domain_x()[ii]};
00383         mtk::Real cc = (north_coefficients_[0])(xx, time);
00384         matrix.SetValue(north_offset + ii, north_offset + ii, cc);
00385     }
00386
00387     if (highest_order_diff_north_ > 0) {
00388 }
00389
00390     return true;
00391 }
00392
00393
00394 }
00395
00396 return true;
00397 }
00398
00399 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace
00400 (
00401     const mtk::Lap2D &lap,
00402     const mtk::UniStgGrid2D &grid,
00403     mtk::DenseMatrix &matrix,
00404     const mtk::Real &time) const {
00405
00406 // For the south-west corner:
00407 auto cc = (west_coefficients_[0])(grid.south_bndy(), time);
00408
00409

```

```

00410 #if MTK_VERBOSE_LEVEL > 2
00411 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00412     matrix.num_cols() << " columns." << std::endl;
00413 std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00414 #endif
00415
00416
00417
00418 mtk::Real harmonic_mean = mtk::kOne/matrix.GetValue(0, 0) +
00419     mtk::kOne/cc;
00420 harmonic_mean = mtk::kTwo/harmonic_mean;
00421
00422 matrix.SetValue(0, 0, harmonic_mean);
00423
00424 int west_offset(grid.num_cells_x() + 1);
00425
00426 auto first_center_y = grid.south_bndy() + grid.delta_y() /
00427     mtk::kTwo;
00428
00429 // For each west entry on the diagonal (west boundary):
00430 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00431     // Evaluate next set spatial coordinates to evaluate the coefficient.
00432     mtk::Real yy = first_center_y + ii*grid.delta_y();
00433     // Evaluate and assign the Dirichlet coefficient.
00434     cc = (west_coefficients_[0])(yy, time);
00435
00436 #if MTK_VERBOSE_LEVEL > 2
00437 std::cout << "Setting at " << west_offset + ii + 1 << ' ' <<
00438     west_offset + ii + 1 << std::endl;
00439 #endif
00440
00441 matrix.SetValue(west_offset + ii + 1, west_offset + ii + 1, cc);
00442
00443 west_offset += grid.num_cells_x() + 1;
00444 }
00445
00446 // For the north-west corner:
00447 cc = (west_coefficients_[0])(grid.north_bndy(), time);
00448
00449 west_offset += grid.num_cells_x() + 1;
00450 int aux{west_offset};
00451 #if MTK_VERBOSE_LEVEL > 2
00452 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00453 #endif
00454
00455 harmonic_mean = mtk::kOne/matrix.GetValue(aux, aux) +
00456     mtk::kOne/cc;
00457 harmonic_mean = mtk::kTwo/harmonic_mean;
00458
00459 matrix.SetValue(aux, aux, harmonic_mean);
00460
00461 if (highest_order_diff_west_ > 0) {
00462 }
00463
00464 return true;
00465 }
00466 }
00467
00468 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace
00469 (
00470     const mtk::Lap2D &lap,
00471     const mtk::UniStgGrid2D &grid,
00472     mtk::DenseMatrix &matrix,
00473     const mtk::Real &time) const {
00474
00475
00476 int west_offset(grid.num_cells_x() + 1);
00477 // For each west entry on the diagonal:
00478 for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00479     // Evaluate next set spatial coordinates to evaluate the coefficient.
00480     mtk::Real yy{grid.discrete_domain_y()[ii]};
00481     // Evaluate and assign the Dirichlet coefficient.
00482     mtk::Real cc = (west_coefficients_[0])(yy, time);
00483     matrix.SetValue(west_offset + ii, west_offset + ii, cc);
00484     west_offset += grid.num_cells_x() + 1;
00485 }
00486
00487 if (highest_order_diff_west_ > 0) {
00488 }
00489
00490 return true;

```

```

00493 }
00494
00495 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace
(
00496     const mtk::Lap2D &lap,
00497     const mtk::UniStgGrid2D &grid,
00498     mtk::DenseMatrix &matrix,
00499     const mtk::Real &time) const {
00500
00502
00503 // For the south-east corner:
00504 auto cc = (east_coefficients_[0])(grid.south_bndy(), time);
00505
00506 int east_offset(grid.num_cells_x() + 1);
00507 #if MTK_VERBOSE_LEVEL > 2
00508 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00509     matrix.num_cols() << " columns." << std::endl;
00510 std::cout << "Setting at " << east_offset << ' ' << east_offset <<
00511     std::endl;
00512 #endif
00513
00514 mtk::Real harmonic_mean =
00515     mtk::kOne/matrix.GetValue(east_offset,east_offset) +
00516     mtk::kOne/cc;
00517 harmonic_mean = mtk::kTwo/harmonic_mean;
00518 matrix.SetValue(east_offset, east_offset, harmonic_mean);
00519
00520 auto first_center_y = grid.south_bndy() + grid.delta_y()/
00521     mtk::kTwo;
00522
00523 // For each east entry on the diagonal (east boundary):
00524 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00525
00526     east_offset += grid.num_cells_x() + 1;
00527
00528     // Evaluate next set spatial coordinates to evaluate the coefficient.
00529     mtk::Real yy = first_center_y + ii*grid.delta_y();
00530     // Evaluate and assign the Dirichlet coefficient.
00531     cc = (east_coefficients_[0])(yy, time);
00532
00533     #if MTK_VERBOSE_LEVEL > 2
00534         std::cout << "Setting at " << east_offset + ii + 1 << ' ' <<
00535             east_offset + ii + 1 << std::endl;
00536     #endif
00537
00538     matrix.SetValue(east_offset + ii + 1, east_offset + ii + 1, cc);
00539 }
00540
00541 // For the north-east corner:
00542 cc = (east_coefficients_[0])(grid.north_bndy(), time);
00543
00544 east_offset += grid.num_cells_x() + 1;
00545 east_offset += grid.num_cells_x() + 1;
00546 int aux{east_offset};
00547 #if MTK_VERBOSE_LEVEL > 2
00548 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00549 #endif
00550
00551 harmonic_mean =
00552     mtk::kOne/matrix.GetValue(aux, aux) + mtk::kOne/cc;
00553 harmonic_mean = mtk::kTwo/harmonic_mean;
00554
00555 matrix.SetValue(aux, aux, harmonic_mean);
00556
00557 if (highest_order_diff_east_ > 0) {
00558 }
00559
00560 return true;
00561 }
00562 }
00563
00564 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace
(
00565     const mtk::Lap2D &lap,
00566     const mtk::UniStgGrid2D &grid,
00567     mtk::DenseMatrix &matrix,
00568     const mtk::Real &time) const {
00569
00570     int east_offset(grid.num_cells_x() + 1);

```

```

00573 // For each west entry on the diagonal:
00574 for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00575     east_offset += grid.num_cells_x() + 1;
00576     // Evaluate next set spatial coordinates to evaluate the coefficient.
00577     mtk::Real yy{grid.discrete_domain_y())[ii]};
00578     // Evaluate and assign the arithmetic mean of Dirichlet coefficients.
00579     mtk::Real cc = (east_coefficients_[0])(yy, time);
00580     matrix.SetValue(east_offset + ii, east_offset + ii, cc);
00581 }
00582
00583 if (highest_order_diff_east_ > 0) {
00584 }
00585
00586 return true;
00587 }
00588 }
00589 }
00590
00591 bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix(
00592     const mtk::Lap2D &lap,
00593     const mtk::UnistgGrid2D &grid,
00594     mtk::DenseMatrix &matrix,
00595     const mtk::Real &time) const {
00596
00597 #ifdef MTK_PERFORM_PREVENTIONS
00598     mtk::Tools::Prevent(highest_order_diff_south_ == -1,
00599                          __FILE__, __LINE__, __func__);
00600     mtk::Tools::Prevent(highest_order_diff_north_ == -1,
00601                          __FILE__, __LINE__, __func__);
00602     mtk::Tools::Prevent(highest_order_diff_west_ == -1,
00603                          __FILE__, __LINE__, __func__);
00604     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00605                          __FILE__, __LINE__, __func__);
00606     mtk::Tools::Prevent(grid.nature() !=
00607                          mtk::FieldNature::SCALAR,
00608                          __FILE__, __LINE__, __func__);
00609     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00610     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00611     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00612     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00613 #endif
00614
00615     bool success{true};
00616
00617     if (!grid.Bound()) {
00618         success = ImposeOnSouthBoundaryNoSpace(lap, grid, matrix, time);
00619         #ifdef MTK_PERFORM_PREVENTIONS
00620             if (!success) {
00621                 return false;
00622             }
00623         #endif
00624         success = ImposeOnNorthBoundaryNoSpace(lap, grid, matrix, time);
00625         #ifdef MTK_PERFORM_PREVENTIONS
00626             if (!success) {
00627                 return false;
00628             }
00629         #endif
00630         success = ImposeOnWestBoundaryNoSpace(lap, grid, matrix, time);
00631         #ifdef MTK_PERFORM_PREVENTIONS
00632             if (!success) {
00633                 return false;
00634             }
00635         #endif
00636         success = ImposeOnEastBoundaryNoSpace(lap, grid, matrix, time);
00637         #ifdef MTK_PERFORM_PREVENTIONS
00638             if (!success) {
00639                 return false;
00640             }
00641         #endif
00642     } else {
00643         success = ImposeOnSouthBoundaryWithSpace(lap, grid, matrix, time);
00644         #ifdef MTK_PERFORM_PREVENTIONS
00645             if (!success) {
00646                 return false;
00647             }
00648         #endif
00649     }
00650     #endif
00651     success = ImposeOnNorthBoundaryWithSpace(lap, grid, matrix, time);
00652     #ifdef MTK_PERFORM_PREVENTIONS
00653         if (!success) {
00654             return false;
00655         }

```

```

00656     #endif
00657     success = ImposeOnWestBoundaryWithSpace(lap, grid, matrix, time);
00658     #ifdef MTK_PERFORM_PREVENTIONS
00659     if (!success) {
00660         return false;
00661     }
00662     #endif
00663     success = ImposeOnEastBoundaryWithSpace(lap, grid, matrix, time);
00664     #ifdef MTK_PERFORM_PREVENTIONS
00665     if (!success) {
00666         return false;
00667     }
00668     #endif
00669 }
00670
00671     return success;
00672 }
00673
00674 void mtk::RobinBCDescriptor2D::ImposeOnGrid(
00675     mtk::UniStgGrid2D &grid,
00676     const mtk::Real &time) const {
00677
00678     #ifdef MTK_PERFORM_PREVENTIONS
00679     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00680     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00681     mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00682     mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00683     mtk::Tools::Prevent(south_condition_ == nullptr,
00684                         __FILE__, __LINE__, __func__);
00685     mtk::Tools::Prevent(north_condition_ == nullptr,
00686                         __FILE__, __LINE__, __func__);
00687     #endif
00688
00689     if (grid.nature() == mtk::FieldNature::SCALAR) {
00690
00691
00692
00693         mtk::Real xx = grid.west_bndy();
00694         (grid.discrete_field())[0] = south_condition_(xx, time);
00695
00696         xx = xx + grid.delta_x()/mtk::kTwo;
00697         // For every point on the south boundary:
00698         for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00699             (grid.discrete_field())[ii + 1] =
00700                 south_condition_(xx + ii*grid.delta_x(), time);
00701         }
00702
00703         xx = grid.east_bndy();
00704         (grid.discrete_field())[grid.num_cells_x() + 1] =
00705             south_condition_(xx, time);
00706
00707         xx = grid.west_bndy();
00708         int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00709         (grid.discrete_field())[north_offset] = north_condition_(xx, time);
00710
00711         xx = xx + grid.delta_x()/mtk::kTwo;
00712         for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00713             (grid.discrete_field())[north_offset + ii + 1] =
00714                 north_condition_(xx + ii*grid.delta_x(), time);
00715         }
00716
00717         xx = grid.east_bndy();
00718         (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00719             north_condition_(xx, time);
00720
00721
00722
00723
00724
00725         mtk::Real yy = grid.south_bndy();
00726         (grid.discrete_field())[0] =
00727             ((grid.discrete_field())[0] + west_condition_(yy, time)) /
00728             mtk::kTwo;
00729
00730         int west_offset{grid.num_cells_x() + 1 + 1};
00731         yy = yy + grid.delta_y()/mtk::kTwo;
00732         for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00733             #if MTK_VERBOSE_LEVEL > 2
00734                 std::cout << "Adding on " << west_offset << "-th position." << std::endl;
00735             #endif
00736             (grid.discrete_field())[west_offset] =
00737                 west_condition_(yy + ii*grid.delta_y(), time);
00738             west_offset += grid.num_cells_x() + 1 + 1;
00739         }
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749 }
```

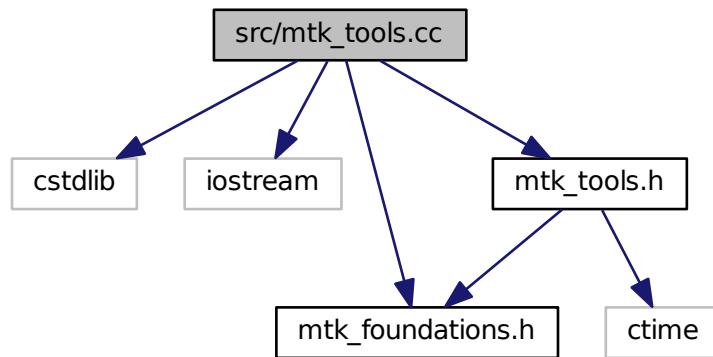
```

00750
00752     yy = grid.north_bndy();
00753     north_offset = (grid.num_cells_y() + 1)*(grid.num_cells_x() + 2);
00754     (grid.discrete_field())[north_offset] =
00755         ((grid.discrete_field())[north_offset] + west_condition_(yy, time))/ 
00756         mtk::kTwo;
00757
00759
00761     yy = grid.south_bndy();
00762     int east_offset{grid.num_cells_x() + 1};
00763     (grid.discrete_field())[east_offset] =
00764         ((grid.discrete_field())[east_offset] + east_condition_(yy, time))/ 
00765         mtk::kTwo;
00766
00768     yy = yy + grid.delta_y()/mtk::kTwo;
00769     for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00770         east_offset += grid.num_cells_x() + 1 + 1;
00771         #if MTK_VERBOSE_LEVEL > 2
00772             std::cout << "Adding on " << east_offset << "-th position." << std::endl;
00773         #endif
00774         (grid.discrete_field())[east_offset] =
00775             east_condition_(yy + ii*grid.delta_y(), time);
00776     }
00777
00779     yy = grid.north_bndy();
00780     (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00781         ((grid.discrete_field())[north_offset + grid.num_cells_x() + 1] + 
00782         east_condition_(yy, time))/mtk::kTwo;
00783
00784 } else {
00785
00787
00789 }
00790 }
```

18.127 src/mtk_tools.cc File Reference

Definition of a class to manage run-time tools.

```
#include <cstdlib>
#include <iostream>
#include "mtk_foundations.h"
#include "mtk_tools.h"
Include dependency graph for mtk_tools.cc:
```



18.127.1 Detailed Description

Definition of a class providing basic tools to ensure execution correctness, and to assist with unitary testing.

Author

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

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

18.128 mtk_tools.cc

```
00001
00011 /*
00012 Copyright (C) 2016, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058
00059 #include <iostream>
00060
00061 #include "mtk_foundations.h"
00062 #include "mtk_tools.h"
00063
00064 void mtk::Tools::Prevent(const bool condition,
00065                         const char *const fname,
00066                         int lineno,
00067                         const char *const fxname) noexcept {
00068
00070 // From: https://www.gnu.org/software/libc/manual/html_node/Normal-Termination.html#Normal-Termination
```

```

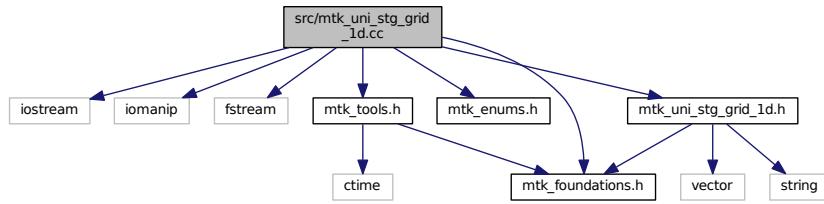
00072 // A process terminates normally when its program signals it is done by
00073 // calling exit. Returning from main is equivalent to calling exit, and the
00074 // value that main returns is used as the argument to exit.
00075
00076 // From: https://www.gnu.org/software/libc/manual/html_node/Exit-Status.html#Exit-Status
00077 // Portability note: Some non-POSIX systems use different conventions for exit
00078 // status values. For greater portability, you can use the macros EXIT_SUCCESS
00079 // and EXIT_FAILURE for the conventional status value for success and failure,
00080 // respectively. They are declared in the file stdlib.h.
00081
00082 if (lineno < 1) {
00083     std::cerr << __FILE__ << ":" << "Incorrect parameter at line " <<
00084     __LINE__ - 2 << "(" << __func__ << ")" << std::endl;
00085     exit(EXIT_FAILURE);
00086 }
00087
00088 if (condition) {
00089     std::cerr << fname << ":" << "Incorrect parameter at line " <<
00090     lineno << "(" << fxname << ")" << std::endl;
00091     exit(EXIT_FAILURE);
00092 }
00093 }
00094
00095 int mtk::Tools::test_number_{}; // Current test being executed.
00096
00097 mtk::Real mtk::Tools::duration_{}; // Duration of the current test.
00098
00099 clock_t mtk::Tools::begin_time_{}; // Elapsed time on current test.
00100
00101 void mtk::Tools::BeginUnitTestNo(const int &nn) noexcept {
00102
00104
00105 #if MTK_PERFORM_PREVENTIONS
00106     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00107 #endif
00108
00109     test_number_ = nn;
00110
00111     std::cout << "Beginning test " << nn << "." << std::endl;
00112     begin_time_ = clock();
00113 }
00114
00115 void mtk::Tools::EndUnitTestNo(const int &nn) noexcept {
00116
00117 #if MTK_PERFORM_PREVENTIONS
00118     mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00119 #endif
00120
00121     duration_ = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00122 }
00123
00124 void mtk::Tools::Assert(const bool &condition) noexcept {
00125
00126     if (condition) {
00127         std::cout << "Test " << test_number_ << ": PASSED in " << duration_ <<
00128         " s." << std::endl;
00129     } else {
00130         std::cout << "Test " << test_number_ << ": FAILED in " << duration_ <<
00131         " s." << std::endl;
00132     }
00133 }
```

18.129 src/mtk_uni_stg_grid_1d.cc File Reference

Implementation of an 1D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include "mtk_foundations.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_1d.h"
```

Include dependency graph for mtk_uni_stg_grid_1d.cc:



Namespaces

- `mtk`

Mimetic Methods Toolkit namespace.

Functions

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

18.129.1 Detailed Description

Implementation of an 1D uniform staggered grid.

Author

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

Definition in file [mtk_uni_stg_grid_1d.cc](#).

18.130 mtk_uni_stg_grid_1d.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
  
```

```

00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include "mtk_foundations.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     stream << "x:";
00075     for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00076         stream << std::setw(10) << in.discrete_domain_x_[ii];
00077     }
00078     stream << std::endl;
00079
00080
00081
00082     if (in.field_nature_ == mtk::FieldNature::SCALAR) {
00083         stream << "u:";
00084     }
00085     else {
00086         stream << "v:";
00087     }
00088     for (unsigned int ii = 0; ii < in.discrete_field_.size(); ++ii) {
00089         stream << std::setw(10) << in.discrete_field_[ii];
00090     }
00091
00092     stream << std::endl;
00093
00094     return stream;
00095 }
00096 }
00097
00098
00099 mtk::UniStgGrid1D& mtk::UniStgGrid1D::operator =(const
00100     mtk::UniStgGrid1D &in) {
00101
00102     if(this == &in) {
00103         return *this;
00104     }
00105     field_nature_ = in.field_nature_;
00106     west_bndy_x_ = in.west_bndy_x_;
00107     east_bndy_x_ = in.east_bndy_x_;
00108     num_cells_x_ = in.num_cells_x_;
00109     delta_x_ = in.delta_x_;
00110
00111     discrete_domain_x_.clear();
00112
00113     std::copy(in.discrete_domain_x_.begin(),

```

```
00114         in.discrete_domain_x_.end(),
00115         discrete_domain_x_.begin());
00116
00117     discrete_field_.clear();
00118
00119     std::copy(in.discrete_field_.begin(),
00120             in.discrete_field_.end(),
00121             discrete_field_.begin());
00122
00123     return *this;
00124 }
00125
00126 mtk::UniStgGrid1D::UniStgGrid1D():
00127     field_nature_(),
00128     discrete_domain_x_(),
00129     discrete_field_(),
00130     west_bndy_x_(),
00131     east_bndy_x_(),
00132     num_cells_x_(),
00133     delta_x_() {}
00134
00135 mtk::UniStgGrid1D::UniStgGrid1D(const
00136 UniStgGrid1D &in):
00137     field_nature_(in.field_nature_),
00138     west_bndy_x_(in.west_bndy_x_),
00139     east_bndy_x_(in.east_bndy_x_),
00140     num_cells_x_(in.num_cells_x_),
00141     delta_x_(in.delta_x_) {
00142
00143     std::copy(in.discrete_domain_x_.begin(),
00144             in.discrete_domain_x_.begin() + in.
00145     discrete_domain_x_.size(),
00146             discrete_domain_x_.begin());
00147
00148     std::copy(in.discrete_field_.begin(),
00149             in.discrete_field_.begin() + in.discrete_field_.size(),
00150             discrete_field_.begin());
00151
00152 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00153                                     const Real &east_bndy_x,
00154                                     const int &num_cells_x,
00155                                     const mtk::FieldNature &field_nature) {
00156
00157 #ifdef MTK_PERFORM_PREVENTIONS
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 #endif
00163
00164     field_nature_ = field_nature;
00165     west_bndy_x_ = west_bndy_x;
00166     east_bndy_x_ = east_bndy_x;
00167     num_cells_x_ = num_cells_x;
00168
00169     delta_x_ = (east_bndy_x - west_bndy_x) / ((mtk::Real) num_cells_x);
00170 }
00171 mtk::UniStgGrid1D::~UniStgGrid1D() {}
00172
00173 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00174
00175     return west_bndy_x_;
00176 }
00177
00178 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00179
00180     return east_bndy_x_;
00181 }
00182
00183 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00184
00185     return delta_x_;
00186 }
00187
00188 const mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() const
00189 {
00190     return discrete_domain_x_.data();
00191 }
```

```

00192
00193 mtk::Real *mtk::UniStgGrid1D::discrete_field() {
00194     return discrete_field_.data();
00195 }
00196
00197 int mtk::UniStgGrid1D::num_cells_x() const {
00198     return num_cells_x_;
00199 }
00200
00201 }
00202
00203 mtk::FieldNature mtk::UniStgGrid1D::field_nature() const {
00204
00205     return field_nature_;
00206 }
00207
00208 void mtk::UniStgGrid1D::GenerateDiscreteDomainX() {
00209
00210 #ifdef MTK_PERFORM_PREVENTIONS
00211     mtk::Tools::Prevent(discrete_domain_x_.size() != 0,
00212                         __FILE__, __LINE__, __func__);
00213 #endif
00214
00215 if (field_nature_ == mtk::FieldNature::SCALAR) {
00216
00217     discrete_domain_x_.reserve(num_cells_x_ + 2);
00218
00219     discrete_domain_x_.push_back(west_bndy_x_);
00220 #ifdef MTK_PRECISION_DOUBLE
00221     auto first_center = west_bndy_x_ + delta_x_/2.0;
00222 #else
00223     auto first_center = west_bndy_x_ + delta_x_/2.0f;
00224 #endif
00225     discrete_domain_x_.push_back(first_center);
00226     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00227         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00228     }
00229     discrete_domain_x_.push_back(east_bndy_x_);
00230
00231 } else {
00232
00233     discrete_domain_x_.reserve(num_cells_x_ + 1);
00234
00235     discrete_domain_x_.push_back(west_bndy_x_);
00236     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00237         discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00238     }
00239     discrete_domain_x_.push_back(east_bndy_x_);
00240
00241 }
00242
00243 void mtk::UniStgGrid1D::ReserveDiscreteField() {
00244
00245 #ifdef MTK_PERFORM_PREVENTIONS
00246     mtk::Tools::Prevent(discrete_field_.size() != 0,
00247                         __FILE__, __LINE__, __func__);
00248 #endif
00249
00250 if (field_nature_ == mtk::FieldNature::SCALAR) {
00251
00252     discrete_field_.reserve(num_cells_x_ + 2);
00253
00254 } else {
00255
00256     discrete_field_.reserve(num_cells_x_ + 1);
00257
00258 }
00259
00260 void mtk::UniStgGrid1D::BindScalarField(
00261     mtk::Real (*ScalarField)(const mtk::Real &xx,
00262                             const std::vector<mtk::Real> &pp),
00263     const std::vector<Real> &parameters) {
00264
00265 #ifdef MTK_PERFORM_PREVENTIONS
00266     mtk::Tools::Prevent(field_nature_ ==
00267                         mtk::FieldNature::VECTOR,
00268                         __FILE__, __LINE__, __func__);
00269 #endif
00270
00271     discrete_domain_x_.reserve(num_cells_x_ + 2);
00272

```

```

00273
00274     discrete_domain_x_.push_back(west_bndy_x_);
00275 #ifdef MTK_PRECISION_DOUBLE
00276     auto first_center = west_bndy_x_ + delta_x_/2.0;
00277 #else
00278     auto first_center = west_bndy_x_ + delta_x_/2.0f;
00279 #endif
00280     discrete_domain_x_.push_back(first_center);
00281     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00282         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00283     }
00284     discrete_domain_x_.push_back(east_bndy_x_);
00285
00286
00287     std::vector<mtk::Real> aux(parameters);
00288
00289     discrete_field_.reserve(num_cells_x_ + 2);
00290
00291     discrete_field_.push_back(ScalarField(west_bndy_x_, aux));
00292
00293     discrete_field_.push_back(ScalarField(first_center, aux));
00294     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00295         discrete_field_.push_back(ScalarField(first_center + ii*delta_x_,
00296                                         aux));
00297     }
00298
00299     discrete_field_.push_back(ScalarField(east_bndy_x_, aux));
00300 }
00301
00302 void mtk::UniStgGrid1D::BindScalarField(const std::vector<Real> &samples)
00303 {
00304 #ifdef MTK_PERFORM_PREVENTIONS
00305     mtk::Tools::Prevent(field_nature_ ==
00306                         mtk::FieldNature::VECTOR,
00307                         __FILE__, __LINE__, __func__);
00308     mtk::Tools::Prevent(samples.size() != num_cells_x_,
00309                         __FILE__, __LINE__, __func__);
00310 #endif
00311     discrete_field_ = samples;
00312 }
00313
00314 void mtk::UniStgGrid1D::BindVectorField(
00315     mtk::Real (*VectorField)(const mtk::Real &xx,
00316                             const std::vector<mtk::Real> &pp),
00317     const std::vector<Real> &parameters) {
00318
00319 #ifdef MTK_PERFORM_PREVENTIONS
00320     mtk::Tools::Prevent(field_nature_ ==
00321                         mtk::FieldNature::SCALAR,
00322                         __FILE__, __LINE__, __func__);
00323 #endif
00324
00325     discrete_domain_x_.reserve(num_cells_x_ + 1);
00326
00327     discrete_domain_x_.push_back(west_bndy_x_);
00328     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00329         discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00330     }
00331     discrete_domain_x_.push_back(east_bndy_x_);
00332
00333
00334     std::vector<mtk::Real> aux(parameters);
00335
00336     discrete_field_.reserve(num_cells_x_ + 1);
00337
00338     discrete_field_.push_back(VectorField(west_bndy_x_, aux));
00339     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00340         discrete_field_.push_back(VectorField(
00341             west_bndy_x_ + ii*delta_x_, aux));
00342     }
00343     discrete_field_.push_back(VectorField(east_bndy_x_, aux));
00344 }
00345
00346
00347 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00348                                     std::string space_name,
00349                                     std::string field_name) const {
00350
00351     std::ofstream output_dat_file; // Output file.
00352
00353

```

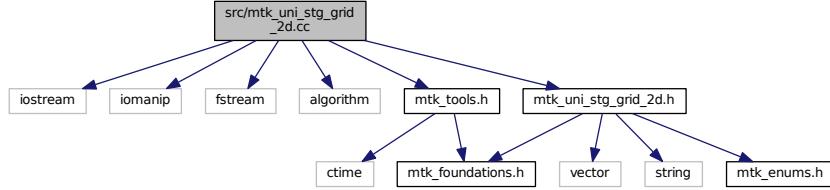
```

00354     output_dat_file.open(filename);
00355
00356     if (!output_dat_file.is_open()) {
00357         return false;
00358     }
00359
00360     output_dat_file << "# " << space_name << ' ' << field_name << std::endl;
00361     for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00362         output_dat_file << discrete_domain_x_[ii] << ' ' << discrete_field_[ii] <<
00363             std::endl;
00364     }
00365
00366     output_dat_file.close();
00367
00368     return true;
00369 }
```

18.131 src/mtk_uni_stg_grid_2d.cc File Reference

Implementation of a 2D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_2d.h"
Include dependency graph for mtk_uni_stg_grid_2d.cc:
```



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

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

18.131.1 Detailed Description

Implementation of a 2D uniform staggered grid.

Author

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

Definition in file [mtk_uni_stg_grid_2d.cc](#).

18.132 mtk_uni_stg_grid_2d.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid2D &in) {
00068
00069 stream << '[' << in.west_bndy_ << ':' << in.num_cells_x_ << ':' <<
00070 in.east_bndy_ << "] x ";
00071
00072 stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073 in.north_bndy_ << "] = " << std::endl << std::endl;
00074
00075
00076
00077 stream << "x:";
```

```

00078     for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00079         stream << std::setw(10) << in.discrete_domain_x_[ii];
00080     }
00081     stream << std::endl;
00082
00083     stream << "y:";
00084     for (unsigned int ii = 0; ii < in.discrete_domain_y_.size(); ++ii) {
00085         stream << std::setw(10) << in.discrete_domain_y_[ii];
00086     }
00087     stream << std::endl;
00088
00089
00090     if (in.nature_ == mtk::FieldNature::SCALAR) {
00091         stream << "u:" << std::endl;
00092         if (in.discrete_field_.size() > 0) {
00093             for (int ii = 0; ii < in.num_cells_x_ + 2; ++ii) {
00094                 for (int jj = 0; jj < in.num_cells_y_ + 2; ++jj) {
00095                     stream << std::setw(10) << in.discrete_field_[ii*in.
00096                                         num_cells_y_ +
00097                                         jj];
00098                 }
00099                 stream << std::endl;
00100             }
00101         }
00102     } else {
00103
00104         int mm{in.num_cells_x_};
00105         int nn{in.num_cells_y_};
00106         int p_offset{nn*(mm + 1) - 1};
00107
00108         stream << "p(x,y):" << std::endl;
00109         for (int ii = 0; ii < nn; ++ii) {
00110             for (int jj = 0; jj < mm + 1; ++jj) {
00111                 stream << std::setw(10) << in.discrete_field_[ii*(mm + 1) + jj];
00112             }
00113             stream << std::endl;
00114         }
00115         stream << std::endl;
00116
00117         stream << "q(x,y):" << std::endl;
00118         for (int ii = 0; ii < nn + 1; ++ii) {
00119             for (int jj = 0; jj < mm; ++jj) {
00120                 stream << std::setw(10) <<
00121                         in.discrete_field_[p_offset + ii*mm + jj];
00122             }
00123             stream << std::endl;
00124         }
00125         stream << std::endl;
00126     }
00127
00128     return stream;
00129 }
00130 }
00131
00132 mtk::UniStgGrid2D::UniStgGrid2D():
00133     discrete_domain_x_(),
00134     discrete_domain_y_(),
00135     discrete_field_(),
00136     nature_(),
00137     west_bndy_(),
00138     east_bndy_(),
00139     num_cells_x_(),
00140     delta_x_(),
00141     south_bndy_(),
00142     north_bndy_(),
00143     num_cells_y_(),
00144     delta_y_() {}
00145
00146 mtk::UniStgGrid2D::UniStgGrid2D(const
00147     UniStgGrid2D &grid):
00148     nature_(grid.nature_),
00149     west_bndy_(grid.west_bndy_),
00150     east_bndy_(grid.east_bndy_),
00151     num_cells_x_(grid.num_cells_x_),
00152     delta_x_(grid.delta_x_),
00153     south_bndy_(grid.south_bndy_),
00154     north_bndy_(grid.north_bndy_),
00155     num_cells_y_(grid.num_cells_y_),
00156     delta_y_(grid.delta_y_) {
00157
00158     std::copy(grid.discrete_domain_x_.begin(),

```

```

00158         grid.discrete_domain_x_.begin() + grid.
00159             discrete_domain_x_.size(),
00160             discrete_domain_x_.begin());
00161     std::copy(grid.discrete_domain_y_.begin(),
00162               grid.discrete_domain_y_.begin() + grid.
00163                   discrete_domain_y_.size(),
00164                   discrete_domain_y_.begin());
00165     std::copy(grid.discrete_field_.begin(),
00166               grid.discrete_field_.begin() + grid.discrete_field_.size(),
00167               discrete_field_.begin());
00168 }
00169
00170 mtk::UniStgGrid2D::UniStgGrid2D(const Real &west_bndy,
00171                                     const Real &east_bndy,
00172                                     const int &num_cells_x,
00173                                     const Real &south_bndy,
00174                                     const Real &north_bndy,
00175                                     const int &num_cells_y,
00176                                     const mtk::FieldNature &nature) {
00177
00178 #ifdef MTK_PERFORM_PREVENTIONS
00179     mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00180     mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00181     mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00182     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00183     mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00184     mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00185     mtk::Tools::Prevent(north_bndy <= south_bndy,
00186                           __FILE__, __LINE__, __func__);
00187     mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00188 #endif
00189
00190     nature_ = nature;
00191
00192     west_bndy_ = west_bndy;
00193     east_bndy_ = east_bndy;
00194     num_cells_x_ = num_cells_x;
00195
00196     south_bndy_ = south_bndy;
00197     north_bndy_ = north_bndy;
00198     num_cells_y_ = num_cells_y;
00199
00200     delta_x_ = (east_bndy_ - west_bndy_) / ((mtk::Real) num_cells_x);
00201     delta_y_ = (north_bndy_ - south_bndy_) / ((mtk::Real) num_cells_y);
00202 }
00203
00204 mtk::UniStgGrid2D::~UniStgGrid2D() {}
00205
00206 mtk::FieldNature mtk::UniStgGrid2D::nature() const {
00207
00208     return nature_;
00209 }
00210
00211 mtk::Real mtk::UniStgGrid2D::west_bndy() const {
00212
00213     return west_bndy_;
00214 }
00215
00216 mtk::Real mtk::UniStgGrid2D::east_bndy() const {
00217
00218     return east_bndy_;
00219 }
00220
00221 int mtk::UniStgGrid2D::num_cells_x() const {
00222
00223     return num_cells_x_;
00224 }
00225
00226 mtk::Real mtk::UniStgGrid2D::delta_x() const {
00227
00228     return delta_x_;
00229 }
00230
00231 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_x() const
00232 {
00233     return discrete_domain_x_.data();
00234 }
00235

```

```

00236 mtk::Real mtk::UniStgGrid2D::south_bndy() const {
00237     return south_bndy_;
00238 }
00239
00240
00241 mtk::Real mtk::UniStgGrid2D::north_bndy() const {
00242     return north_bndy_;
00243 }
00244
00245
00246 int mtk::UniStgGrid2D::num_cells_y() const {
00247     return num_cells_y_;
00248 }
00249
00250
00251 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00252     return delta_y_;
00253 }
00254
00255
00256 bool mtk::UniStgGrid2D::Bound() const {
00257     return discrete_field_.size() != 0;
00258 }
00259
00260
00261 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_y() const
{
00262
00263     return discrete_domain_y_.data();
00264 }
00265
00266 mtk::Real* mtk::UniStgGrid2D::discrete_field() {
00267
00268     return discrete_field_.data();
00269 }
00270
00271 int mtk::UniStgGrid2D::Size() const {
00272
00273     return discrete_field_.size();
00274 }
00275
00276 void mtk::UniStgGrid2D::BindScalarField(
00277     Real (*ScalarField)(const Real &xx, const Real &yy)) {
00278
00279     #ifdef MTK_PERFORM_PREVENTIONS
00280         mtk::Tools::Prevent(nature_ != mtk::FieldNature::SCALAR,
00281         __FILE__, __LINE__,
00282         __func__);
00283     #endif
00284
00285
00286     discrete_domain_x_.reserve(num_cells_x_ + 2);
00287
00288     discrete_domain_x_.push_back(west_bndy_);
00289     #ifdef MTK_PRECISION_DOUBLE
00290         auto first_center = west_bndy_ + delta_x_/2.0;
00291     #else
00292         auto first_center = west_bndy_ + delta_x_/2.0f;
00293     #endif
00294     discrete_domain_x_.push_back(first_center);
00295     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00296         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00297     }
00298     discrete_domain_x_.push_back(east_bndy_);
00299
00300
00301     discrete_domain_y_.reserve(num_cells_y_ + 2);
00302
00303     discrete_domain_y_.push_back(south_bndy_);
00304     #ifdef MTK_PRECISION_DOUBLE
00305         first_center = south_bndy_ + delta_x_/2.0;
00306     #else
00307         first_center = south_bndy_ + delta_x_/2.0f;
00308     #endif
00309     discrete_domain_y_.push_back(first_center);
00310     for (auto ii = 1; ii < num_cells_y_; ++ii) {
00311         discrete_domain_y_.push_back(first_center + ii*delta_y_);
00312     }
00313     discrete_domain_y_.push_back(north_bndy_);
00314
00315
00316
00317

```

```

00318 discrete_field_.reserve((num_cells_x_ + 2)*(num_cells_y_ + 2));
00319
00320 for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00321     for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00322         #if MTK_VERBOSE_LEVEL > 6
00323             std::cout << "Pushing value for x = " << discrete_domain_x_[jj] <<
00324                 " y = " << discrete_domain_y_[ii] << std::endl;
00325         #endif
00326         discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00327                                             discrete_domain_y_[ii]));
00328     }
00329 }
00330
00331
00332 void mtk::UniStgGrid2D::BindVectorFieldPComponent(
00333     mtk::Real (*VectorField)(const mtk::Real &xx, const
00334     mtk::Real &yy)) {
00335     int mm{num_cells_x_};
00336     int nn{num_cells_y_};
00337
00338     int total{nn*(mm + 1) + mm*(nn + 1)};
00339
00340 #ifdef MTK_PRECISION_DOUBLE
00341     double half_delta_x{delta_x_/2.0};
00342     double half_delta_y{delta_y_/2.0};
00343 #else
00344     float half_delta_x{delta_x_/2.0f};
00345     float half_delta_y{delta_y_/2.0f};
00346 #endif
00347
00348
00349 // We need every data point of the discrete domain; i.e. we need all the
00350 // nodes and all the centers. There are mm centers for the x direction, and
00351 // nn centers for the y direction. Since there is one node per center, that
00352 // amounts to 2*mm. If we finally consider the final boundary node, it
00353 // amounts to a total of 2*mm + 1 for the x direction. Analogously, for the
00354 // y direction, this amounts to 2*nn + 1.
00355
00356     discrete_domain_x_.reserve(2*mm + 1);
00357
00358     discrete_domain_x_.push_back(west_bndy_);
00359     for (int ii = 1; ii < (2*mm + 1); ++ii) {
00360         discrete_domain_x_.push_back(west_bndy_ + ii*half_delta_x);
00361     }
00362
00363
00364     discrete_domain_y_.reserve(2*nn + 1);
00365
00366     discrete_domain_y_.push_back(south_bndy_);
00367     for (int ii = 1; ii < (2*nn + 1); ++ii) {
00368         discrete_domain_y_.push_back(south_bndy_ + ii*half_delta_y);
00369     }
00370
00371
00372     discrete_field_.reserve(total);
00373
00374 // For each y-center.
00375     for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00376
00377 // Bind all of the x-nodes for this y-center.
00378     for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00379         discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00380                                             discrete_domain_y_[ii]));
00381
00382         #if MTK_VERBOSE_LEVEL > 6
00383             std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00384                 discrete_domain_y_[ii] << " = " <<
00385                 VectorField(discrete_domain_x_[jj], discrete_domain_y_[ii]) << std::endl;
00386         #endif
00387     }
00388
00389 }
00390
00391 }
00392 #if MTK_VERBOSE_LEVEL > 6
00393     std::cout << std::endl;
00394 #endif
00395 }
00396
00397 void mtk::UniStgGrid2D::BindVectorFieldQComponent(
00398     mtk::Real (*VectorField)(const mtk::Real &xx, const
00399     mtk::Real &yy)) {
00400

```

```

00400     int mm{num_cells_x_};
00401     int nn{num_cells_y_};
00402
00403
00404
00405 // For each y-node.
00406 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00407
00408     // Bind all of the x-center for this y-node.
00409     for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00410         discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00411                                             discrete_domain_y_[ii]));
00412
00413         #if MTK_VERBOSE_LEVEL > 6
00414             std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00415                 discrete_domain_y_[ii] << " = " <<
00416                 VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00417         #endif
00418     }
00419 }
00420 #if MTK_VERBOSE_LEVEL > 6
00421 std::cout << std::endl;
00422 #endif
00423 }
00424
00425 void mtk::UniStgGrid2D::BindVectorField(
00426     Real (*VectorFieldPComponent)(const Real &xx, const
00427     Real &yy),
00428     Real (*VectorFieldQComponent)(const Real &xx, const
00429     Real &yy)) {
00430
00431     #ifdef MTK_PERFORM_PREVENTIONS
00432         mtk::Tools::Prevent(nature_ != mtk::FieldNature::VECTOR,
00433         __FILE__, __LINE__,
00434         __func__);
00435     #endif
00436
00437     BindVectorFieldPComponent(VectorFieldPComponent);
00438     BindVectorFieldQComponent(VectorFieldQComponent);
00439 }
00440
00441 bool mtk::UniStgGrid2D::WriteToFile(std::string filename,
00442                                         std::string space_name_x,
00443                                         std::string space_name_y,
00444                                         std::string field_name) const {
00445
00446     std::ofstream output_dat_file; // Output file.
00447
00448     output_dat_file.open(filename);
00449
00450     if (!output_dat_file.is_open())
00451         return false;
00452
00453     if (nature_ == mtk::FieldNature::SCALAR) {
00454         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00455         field_name << std::endl;
00456
00457         int idx{};
00458         for (unsigned int ii = 0; ii < discrete_domain_y_.size(); ++ii) {
00459             for (unsigned int jj = 0; jj < discrete_domain_x_.size(); ++jj) {
00460                 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00461                     discrete_domain_y_[ii] << ' ' <<
00462                     discrete_field_[idx] <<
00463                         std::endl;
00464             }
00465             output_dat_file << std::endl;
00466         }
00467     } else {
00468         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00469         field_name << std::endl;
00470
00471         output_dat_file << "# Horizontal component:" << std::endl;
00472
00473         int mm{num_cells_x_};
00474         int nn{num_cells_y_};
00475
00476
00477 // For each y-center.
00478         int idx{};
00479         for (int ii = 1; ii < 2*nn + 1; ii += 2) {

```

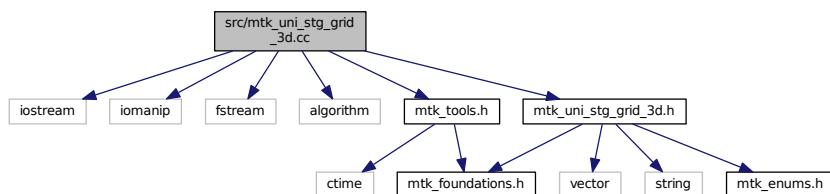
```

00480 // Bind all of the x-nodes for this y-center.
00481 for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00482
00483     output_dat_file << discrete_domain_x_[jj] << ' ' <<
00484         discrete_domain_y_[ii] << ' ' << discrete_field_[idx] << ' ' <<
00485         mtk::kZero << std::endl;
00486
00487     ++idx;
00488 }
00489
00490
00491     int p_offset{nn*(mm + 1) - 1};
00492     idx = 0;
00493     output_dat_file << "# Vertical component:" << std::endl;
00495 // For each y-node.
00496 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00497
00498     // Bind all of the x-center for this y-node.
00499     for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00500
00501         output_dat_file << discrete_domain_x_[jj] << ' ' <<
00502             discrete_domain_y_[ii] << ' ' << mtk::kZero << ' ' <<
00503             discrete_field_[p_offset + idx] << std::endl;
00504
00505     ++idx;
00506 }
00507 }
00508
00509 output_dat_file.close();
00510
00511 return true;
00512 }
```

18.133 src/mtk_uni_stg_grid_3d.cc File Reference

Implementation of a 3D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_3d.h"
Include dependency graph for mtk_uni_stg_grid_3d.cc:
```



Namespaces

- **mtk**

Mimetic Methods Toolkit namespace.

Functions

- std::ostream & [mtk::operator<<](#) (std::ostream &stream, [mtk::UniStgGrid3D](#) &in)

18.133.1 Detailed Description

Implementation of a 3D uniform staggered grid.

Author

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

Definition in file [mtk_uni_stg_grid_3d.cc](#).

18.134 mtk_uni_stg_grid_3d.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"

```

```
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid3D &in) {
00068
00069     stream << '[' << in.west_bndy_ << ':' << in.num_cells_x_ << ':' <<
00070     in.east_bndy_ << "] x ";
00071
00072     stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073     in.north_bndy_ << "] x ";
00074
00075     stream << '[' << in.bottom_bndy_ << ':' << in.num_cells_z_ << ':' <<
00076     in.top_bndy_ << "] = " << std::endl << std::endl;
00077
00078
00079     stream << "x:";
00080     for (auto const &cc: in.discrete_domain_x_) {
00081         stream << std::setw(10) << cc;
00082     }
00083     stream << std::endl;
00084
00085     stream << "y:";
00086     for (auto const &cc: in.discrete_domain_y_) {
00087         stream << std::setw(10) << cc;
00088     }
00089     stream << std::endl;
00090
00091     stream << "z:";
00092     for (auto const &cc: in.discrete_domain_z_) {
00093         stream << std::setw(10) << cc;
00094     }
00095     stream << std::endl;
00096
00097
00098     if (in.nature_ == mtk::FieldNature::SCALAR) {
00099         stream << "u(x,y,z):" << std::endl;
00100         if (in.discrete_field_.size() > 0) {
00101             }
00102         } else {
00103             stream << "p(x,y,z):" << std::endl;
00104             stream << "q(x,y,z):" << std::endl;
00105             if (in.discrete_field_.size() > 0) {
00106                 }
00107             }
00108         }
00109     }
00110     return stream;
00111 }
00112
00113 }
00114
00115 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator=(const
00116     mtk::UniStgGrid3D &in) {
00117
00118     UniStgGrid3D out(in);
00119
00120     return out;
00121 }
00122
00123 mtk::UniStgGrid3D::UniStgGrid3D():
00124     discrete_domain_x_(),
00125     discrete_domain_y_(),
00126     discrete_domain_z_(),
00127     discrete_field_(),
00128     nature_(),
00129     west_bndy_(),
00130     east_bndy_(),
00131     num_cells_x_(),
00132     delta_x_(),
00133     south_bndy_(),
00134     north_bndy_(),
00135     num_cells_y_(),
00136     delta_y_(),
00137     bottom_bndy_(),
00138     top_bndy_(),
00139     num_cells_z_(),
00140     delta_z_() {}
00141
00142 mtk::UniStgGrid3D::UniStgGrid3D(const
00143     UniStgGrid3D &grid):
00144     nature_(grid.nature_),
```

```

00144     west_bndy_(grid.west_bndy_),
00145     east_bndy_(grid.east_bndy_),
00146     num_cells_x_(grid.num_cells_x_),
00147     delta_x_(grid.delta_x_),
00148     south_bndy_(grid.south_bndy_),
00149     north_bndy_(grid.north_bndy_),
00150     num_cells_y_(grid.num_cells_y_),
00151     delta_y_(grid.delta_y_),
00152     bottom_bndy_(grid.bottom_bndy_),
00153     top_bndy_(grid.top_bndy_),
00154     num_cells_z_(grid.num_cells_z_),
00155     delta_z_(grid.delta_z_) {
00156
00157     std::copy(grid.discrete_domain_x_.begin(),
00158               grid.discrete_domain_x_.begin() + grid.
00159               discrete_domain_x_.size(),
00160               discrete_domain_x_.begin());
00161
00162     std::copy(grid.discrete_domain_y_.begin(),
00163               grid.discrete_domain_y_.begin() + grid.
00164               discrete_domain_y_.size(),
00165               discrete_domain_y_.begin());
00166
00167     std::copy(grid.discrete_domain_z_.begin(),
00168               grid.discrete_domain_z_.begin() + grid.
00169               discrete_domain_z_.size(),
00170               discrete_domain_z_.begin());
00171
00172 }
00173
00174 mtk::UniStgGrid3D::UniStgGrid3D(const Real &west_bndy,
00175                                     const Real &east_bndy,
00176                                     const int &num_cells_x,
00177                                     const Real &south_bndy,
00178                                     const Real &north_bndy,
00179                                     const int &num_cells_y,
00180                                     const Real &bottom_bndy,
00181                                     const Real &top_bndy,
00182                                     const int &num_cells_z,
00183                                     const mtk::FieldNature &nature) {
00184
00185 #ifdef MTK_PERFORM_PREVENTIONS
00186     mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00187     mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00188     mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00189     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00190     mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00191     mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00192     mtk::Tools::Prevent(north_bndy <= south_bndy,
00193                         __FILE__, __LINE__, __func__);
00194     mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00195     mtk::Tools::Prevent(bottom_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00196     mtk::Tools::Prevent(top_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00197     mtk::Tools::Prevent(top_bndy <= bottom_bndy,
00198                         __FILE__, __LINE__, __func__);
00199     mtk::Tools::Prevent(num_cells_z < 0, __FILE__, __LINE__, __func__);
00200 #endif
00201
00202     nature_ = nature;
00203
00204     west_bndy_ = west_bndy;
00205     east_bndy_ = east_bndy;
00206     num_cells_x_ = num_cells_x;
00207
00208     south_bndy_ = south_bndy;
00209     north_bndy_ = north_bndy;
00210     num_cells_y_ = num_cells_y;
00211
00212     bottom_bndy_ = bottom_bndy;
00213     top_bndy_ = top_bndy;
00214     num_cells_z_ = num_cells_z;
00215
00216     delta_x_ = (east_bndy_ - west_bndy_) / ((mtk::Real) num_cells_x);
00217     delta_y_ = (north_bndy_ - south_bndy_) / ((mtk::Real) num_cells_y);
00218     delta_z_ = (top_bndy_ - bottom_bndy_) / ((mtk::Real) num_cells_z);
00219 }
00220
00221 mtk::UniStgGrid3D::~UniStgGrid3D() {}

```

```
00222
00223 mtk::FieldNature mtk::UniStgGrid3D::nature() const {
00224     return nature_;
00225 }
00226
00227 mtk::Real mtk::UniStgGrid3D::west_bndy() const {
00228     return west_bndy_;
00229 }
00230
00231 mtk::Real mtk::UniStgGrid3D::east_bndy() const {
00232     return east_bndy_;
00233 }
00234
00235 mtk::Real mtk::UniStgGrid3D::delta_x() const {
00236     return delta_x_;
00237 }
00238 int mtk::UniStgGrid3D::num_cells_x() const {
00239     return num_cells_x_;
00240 }
00241
00242 mtk::Real mtk::UniStgGrid3D::discrete_domain_x() const {
00243     return discrete_domain_x_.data();
00244 }
00245
00246 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_x() const {
00247     return discrete_domain_x_.data();
00248 }
00249
00250 mtk::Real mtk::UniStgGrid3D::south_bndy() const {
00251     return south_bndy_;
00252 }
00253
00254 mtk::Real mtk::UniStgGrid3D::north_bndy() const {
00255     return north_bndy_;
00256 }
00257
00258 int mtk::UniStgGrid3D::num_cells_y() const {
00259     return num_cells_y_;
00260 }
00261
00262 mtk::Real mtk::UniStgGrid3D::delta_y() const {
00263     return delta_y_;
00264 }
00265
00266 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_y() const {
00267     return discrete_domain_y_.data();
00268 }
00269
00270 mtk::Real mtk::UniStgGrid3D::bottom_bndy() const {
00271     return bottom_bndy_;
00272 }
00273
00274 mtk::Real mtk::UniStgGrid3D::top_bndy() const {
00275     return top_bndy_;
00276 }
00277
00278 int mtk::UniStgGrid3D::num_cells_z() const {
00279     return num_cells_z_;
00280 }
00281
00282 mtk::Real mtk::UniStgGrid3D::delta_z() const {
00283     return delta_z_;
00284 }
00285
00286 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_z() const {
00287     return discrete_domain_z_.data();
00288 }
```

```

00300     return discrete_domain_z_.data();
00301 }
00302
00303 mtk::Real* mtk::UniStgGrid3D::discrete_field() {
00304
00305     return discrete_field_.data();
00306 }
00307
00308 bool mtk::UniStgGrid3D::Bound() const {
00309
00310     return discrete_field_.size() != 0;
00311 }
00312
00313 int mtk::UniStgGrid3D::Size() const {
00314
00315     return discrete_field_.size();
00316 }
00317
00318 void mtk::UniStgGrid3D::BindScalarField(
00319     mtk::Real (*ScalarField)(const mtk::Real &xx,
00320                             const mtk::Real &yy,
00321                             const mtk::Real &zz)) {
00322
00323 #ifdef MTK_PERFORM_PREVENTIONS
00324     mtk::Tools::Prevent(nature_ != mtk::FieldNature::SCALAR,
00325     __FILE__, __LINE__,
00326     __func__);
00327 #endif
00328
00329     discrete_domain_x_.reserve(num_cells_x_ + 2);
00330
00331     discrete_domain_x_.push_back(west_bndy_);
00332 #ifdef MTK_PRECISION_DOUBLE
00333     auto first_center = west_bndy_ + delta_x_/2.0;
00334 #else
00335     auto first_center = west_bndy_ + delta_x_/2.0f;
00336 #endif
00337     discrete_domain_x_.push_back(first_center);
00338     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00339         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00340     }
00341     discrete_domain_x_.push_back(east_bndy_);
00342
00343
00344     discrete_domain_y_.reserve(num_cells_y_ + 2);
00345
00346     discrete_domain_y_.push_back(south_bndy_);
00347 #ifdef MTK_PRECISION_DOUBLE
00348     first_center = south_bndy_ + delta_x_/2.0;
00349 #else
00350     first_center = south_bndy_ + delta_x_/2.0f;
00351 #endif
00352     discrete_domain_y_.push_back(first_center);
00353     for (auto ii = 1; ii < num_cells_y_; ++ii) {
00354         discrete_domain_y_.push_back(first_center + ii*delta_y_);
00355     }
00356     discrete_domain_y_.push_back(north_bndy_);
00357
00358
00359     discrete_domain_z_.reserve(num_cells_z_ + 2);
00360
00361     discrete_domain_z_.push_back(bottom_bndy_);
00362 #ifdef MTK_PRECISION_DOUBLE
00363     first_center = bottom_bndy_ + delta_z_/_mtk::kTwo;
00364 #else
00365     first_center = bottom_bndy_ + delta_z_/_mtk::kTwo;
00366 #endif
00367     discrete_domain_z_.push_back(first_center);
00368     for (auto ii = 1; ii < num_cells_z_; ++ii) {
00369         discrete_domain_z_.push_back(first_center + ii*delta_z_);
00370     }
00371     discrete_domain_z_.push_back(top_bndy_);
00372
00373
00374     int aux{ (num_cells_x_ + 2)*(num_cells_y_ + 2)*(num_cells_z_ + 2) };
00375
00376     discrete_field_.reserve(aux);
00377
00378     for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00379         for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00380             for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00381                 #if MTK_VERBOSE_LEVEL > 6
00382                     std::cout << "At z = " << discrete_domain_z_[kk] << ": Pushing value"
00383                     " for x = " << discrete_domain_x_[jj] << " y = " <<
00384

```

```

00384         discrete_domain_y_[ii] << std::endl;
00385     #endif
00386     discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00387                                             discrete_domain_y_[ii],
00388                                             discrete_domain_z_[kk]));
00389   }
00390 }
00391 }
00392 }
00393
00394 void mtk::UniStgGrid3D::BindVectorFieldPComponent(
00395   mtk::Real (*VectorField)(const mtk::Real &xx,
00396                           const mtk::Real &yy,
00397                           const mtk::Real &zz)) {
00398
00399 }
00400
00401 void mtk::UniStgGrid3D::BindVectorFieldQComponent(
00402   mtk::Real (*VectorField)(const mtk::Real &xx,
00403                           const mtk::Real &yy,
00404                           const mtk::Real &zz)) {
00405
00406 }
00407
00408 void mtk::UniStgGrid3D::BindVectorFieldRComponent(
00409   mtk::Real (*VectorField)(const mtk::Real &xx,
00410                           const mtk::Real &yy,
00411                           const mtk::Real &zz)) {
00412
00413 }
00414
00415 void mtk::UniStgGrid3D::BindVectorField(
00416   mtk::Real (*VectorFieldPComponent)(const
00417                                     mtk::Real &xx,
00418                                     const mtk::Real &yy,
00419                                     const mtk::Real &zz),
00420   mtk::Real (*VectorFieldQComponent)(const
00421                                     mtk::Real &xx,
00422                                     const mtk::Real &yy,
00423                                     const mtk::Real &zz),
00424   mtk::Real (*VectorFieldRComponent)(const mtk::Real &xx,
00425                                     const mtk::Real &yy,
00426                                     const mtk::Real &zz)) {
00427
00428 #ifdef MTK_PERFORM_PREVENTIONS
00429   mtk::Tools::Prevent(nature_ != mtk::FieldNature::VECTOR,
00430   __FILE__, __LINE__,
00431   __func__);
00432 #endif
00433
00434 BindVectorFieldPComponent(VectorFieldPComponent);
00435 BindVectorFieldQComponent(VectorFieldQComponent);
00436
00437 bool mtk::UniStgGrid3D::WriteToFile(std::string filename,
00438                                     std::string space_name_x,
00439                                     std::string space_name_y,
00440                                     std::string space_name_z,
00441                                     std::string field_name) const {
00442
00443   std::ofstream output_dat_file; // Output file.
00444
00445   output_dat_file.open(filename);
00446
00447   if (!output_dat_file.is_open()) {
00448     return false;
00449   }
00450
00451   if (nature_ == mtk::FieldNature::SCALAR) {
00452     output_dat_file << "#" << space_name_x << ' ' << space_name_y << ' ' <<
00453     space_name_z << ' ' << field_name << std::endl;
00454
00455   int idx{};
00456   for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00457     for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00458       for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00459         output_dat_file << discrete_domain_x_[jj] << ' ' <<
00460         discrete_domain_y_[ii] << ' ' << discrete_domain_z_[kk] << ' ' <<
00461         discrete_field_[idx] << std::endl;
00462       idx++;
00463     }
00464   }
00465 }
```

```

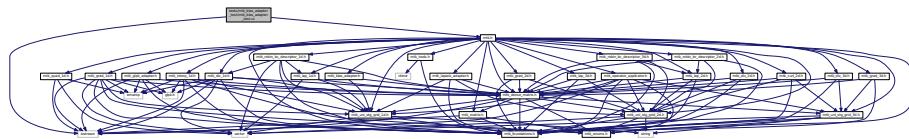
00462     }
00463 }
00464 } else {
00465     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00466     space_name_z << ' ' << field_name << std::endl;
00467 }
00468 }
00469 }
00470 output_dat_file.close();
00471
00472 return true;
00473 }
00474 }
```

18.135 tests/mtk blas adapter test/mtk blas adapter test.cc File Reference

Test file for the [mtk::BLASAdapter](#) class.

```
#include <iostream>
#include "mtk.h"
```

Include dependency graph for mtk blas adapter test.cc:



Functions

- void [TestRealDenseMM \(\)](#)
- int [main \(\)](#)

18.135.1 Detailed Description

Author

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

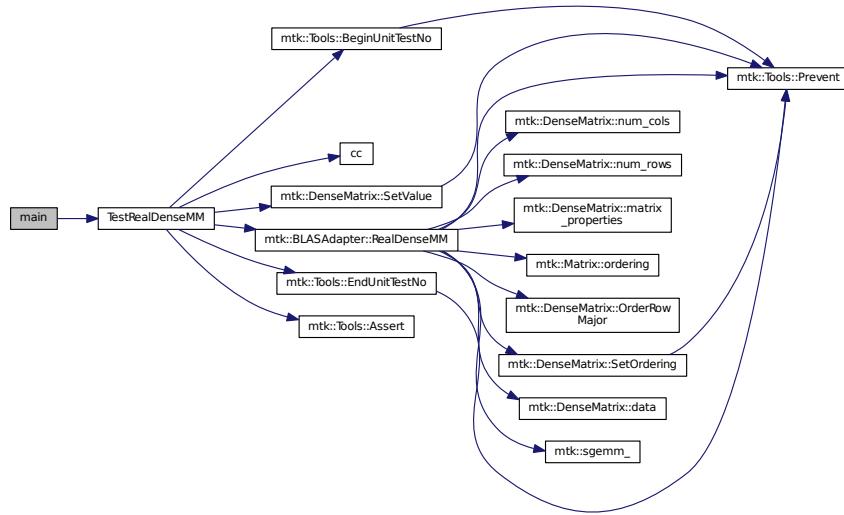
Definition in file [mtk blas adapter test.cc](#).

18.135.2 Function Documentation

18.135.2.1 int main ()

Definition at line [96](#) of file [mtk blas adapter test.cc](#).

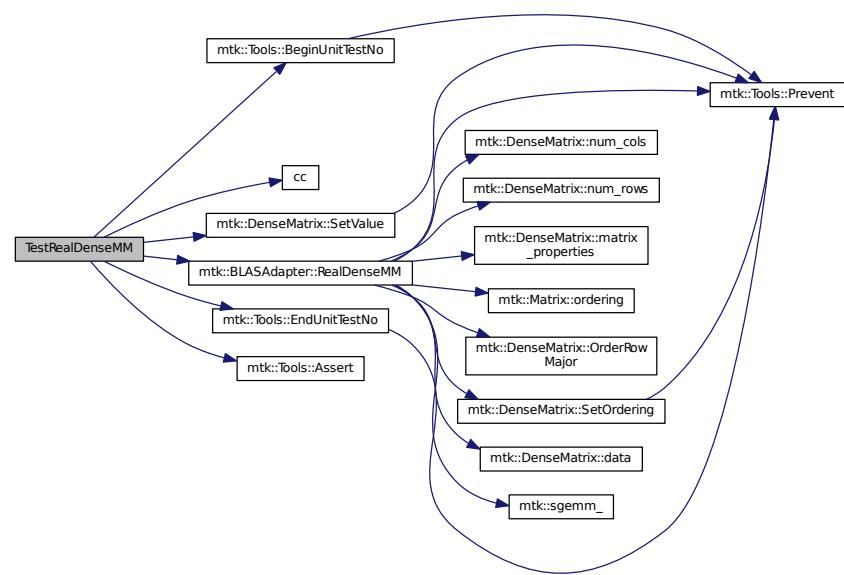
Here is the call graph for this function:



18.135.2.2 void TestRealDenseMM ()

Definition at line 58 of file [mtk blas_adapter_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.136 mtk_blas_adapter_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <iostream>
00055
00056 #include "mtk.h"
00057
00058 void TestRealDenseMM() {
00059
00060     mtk::Tools::BeginUnitTestNo(1);
00061
00062     int rr = 2;
00063     int cc = 3;
  
```

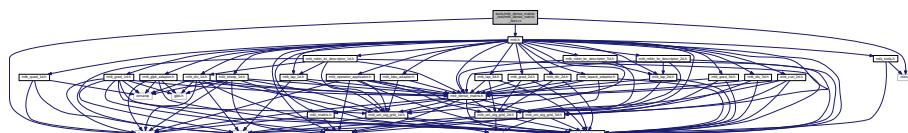
```

00064
00065     mtk::DenseMatrix aa(rr,cc);
00066
00067     aa.SetValue(0,0,1.0);
00068     aa.SetValue(0,1,2.0);
00069     aa.SetValue(0,2,3.0);
00070     aa.SetValue(1,0,4.0);
00071     aa.SetValue(1,1,5.0);
00072     aa.SetValue(1,2,6.0);
00073
00074     mtk::DenseMatrix bb(cc,rr);
00075
00076     bb.SetValue(0,0,7.0);
00077     bb.SetValue(0,1,8.0);
00078     bb.SetValue(1,0,9.0);
00079     bb.SetValue(1,1,10.0);
00080     bb.SetValue(2,0,11.0);
00081     bb.SetValue(2,1,12.0);
00082
00083     mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00084
00085     mtk::DenseMatrix ff(rr,rr);
00086
00087     ff.SetValue(0,0,58.0);
00088     ff.SetValue(0,1,64.00);
00089     ff.SetValue(1,0,139.0);
00090     ff.SetValue(1,1,154.0);
00091
00092     mtk::Tools::EndUnitTestNo(1);
00093     mtk::Tools::Assert(pp == ff);
00094 }
00095
00096 int main () {
00097
00098     std::cout << "Testing mtk::BLASAdapter class." << std::endl;
00099
00100     TestRealDenseMM();
00101 }
```

18.137 tests/mtk_dense_matrix_test/mtk_dense_matrix_test.cc File Reference

Test file for the `mtk::DenseMatrix` class.

```
#include <iostream>
#include <ctime>
#include "mtk.h"
Include dependency graph for mtk_dense_matrix_test.cc:
```



Functions

- void `TestDefaultConstructor ()`
- void `TestConstructorWithNumRowsNumCols ()`
- void `TestConstructAsIdentity ()`
- void `TestConstructAsVandermonde ()`
- void `TestSetValueGetValue ()`
- void `TestConstructAsVandermondeTranspose ()`
- void `TestKron ()`

- void [TestConstructWithNumRowsNumColsAssignmentOperator \(\)](#)
- void [TestConstructAsVandermondeTransposeAssignmentOperator \(\)](#)
- int [main \(\)](#)

18.137.1 Detailed Description

Author

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

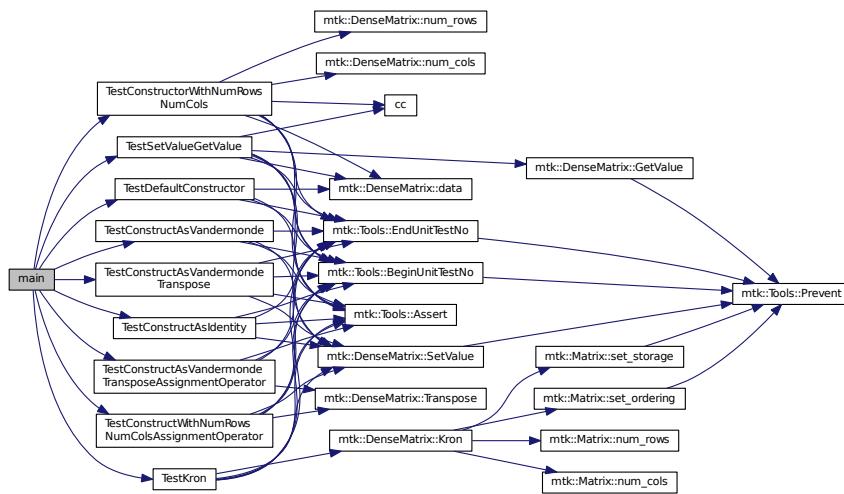
Definition in file [mtk_dense_matrix_test.cc](#).

18.137.2 Function Documentation

18.137.2.1 int main ()

Definition at line 309 of file [mtk_dense_matrix_test.cc](#).

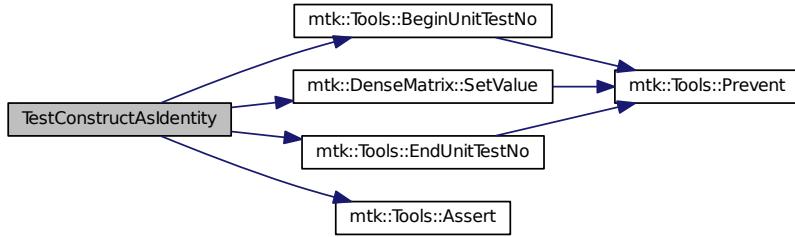
Here is the call graph for this function:



18.137.2.2 void TestConstructAsIdentity ()

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

Here is the call graph for this function:



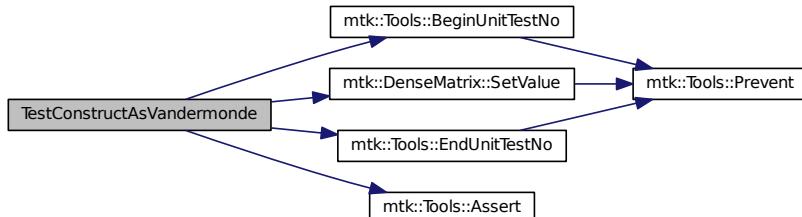
Here is the caller graph for this function:



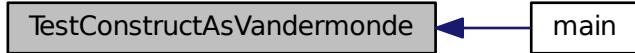
18.137.2.3 void TestConstructAsVandermonde ()

Definition at line 106 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



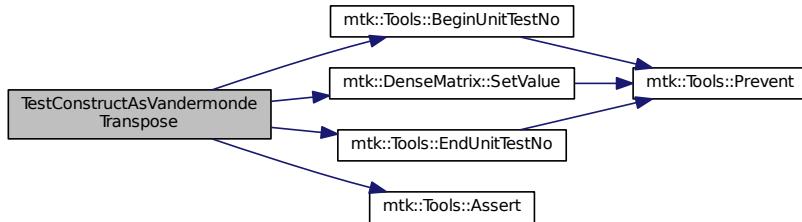
Here is the caller graph for this function:



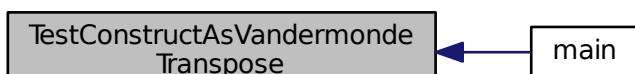
18.137.2.4 void TestConstructAsVandermondeTranspose ()

Definition at line 155 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



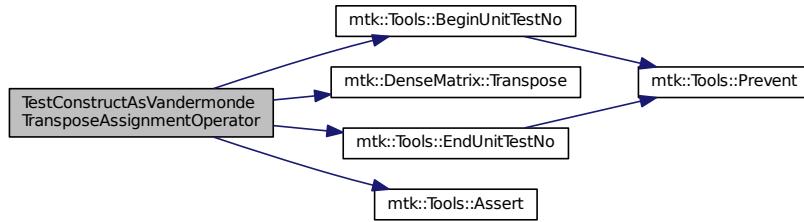
Here is the caller graph for this function:



18.137.2.5 void TestConstructAsVandermondeTransposeAssignmentOperator ()

Definition at line 286 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



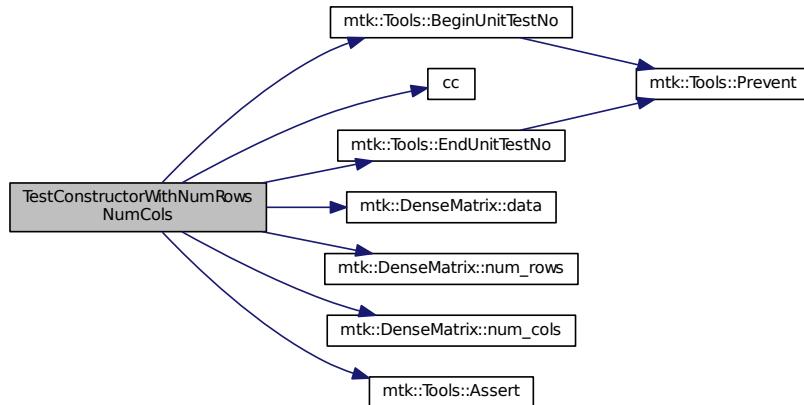
Here is the caller graph for this function:



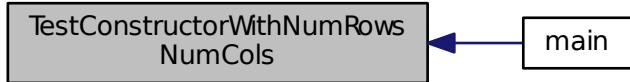
18.137.2.6 void TestConstructorWithNumRowsNumCols()

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

Here is the call graph for this function:



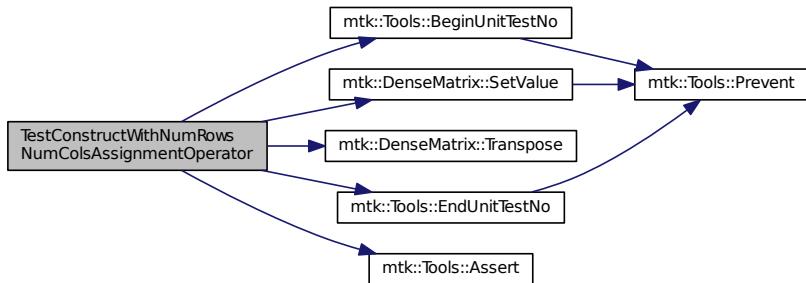
Here is the caller graph for this function:



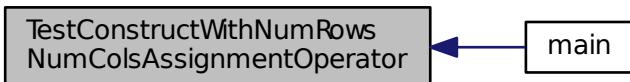
18.137.2.7 void TestConstructWithNumRowsNumColsAssignmentOperator ()

Definition at line 262 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



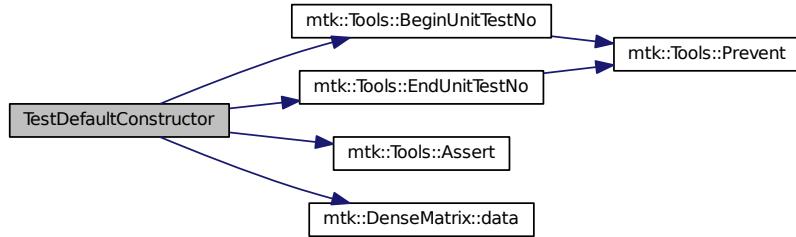
Here is the caller graph for this function:



18.137.2.8 void TestDefaultConstructor ()

Definition at line 59 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



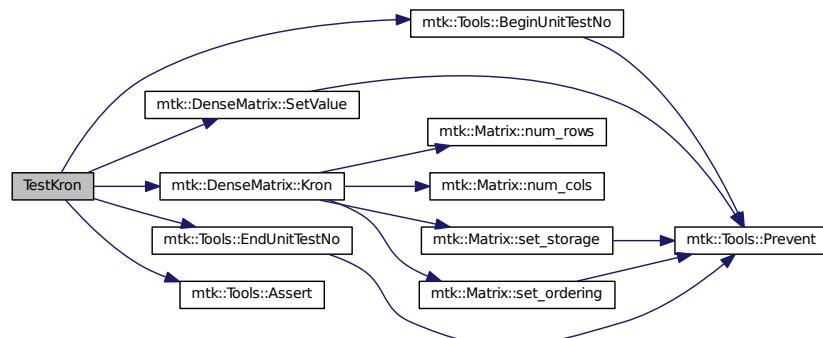
Here is the caller graph for this function:



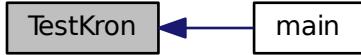
18.137.2.9 void TestKron ()

Definition at line 192 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



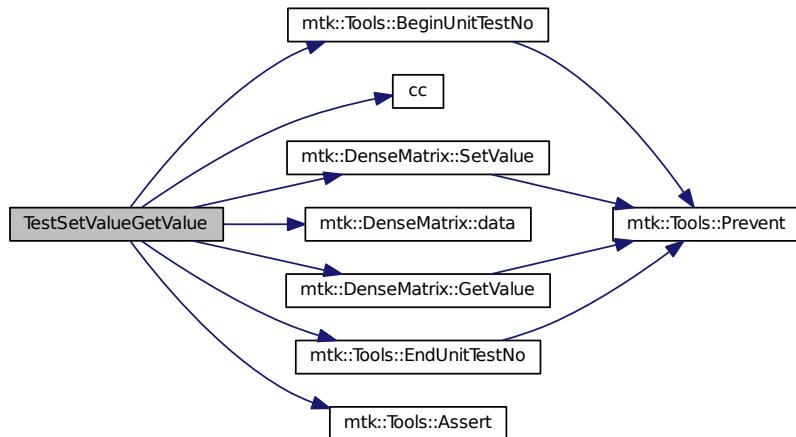
Here is the caller graph for this function:



18.137.2.10 void TestSetValueGetValue()

Definition at line 126 of file [mtk_dense_matrix_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.138 mtk_dense_matrix_test.cc

```
00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
0010 University. All rights reserved.
0011
0012 Redistribution and use in source and binary forms, with or without modification,
0013 are permitted provided that the following conditions are met:
0014
0015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
0016 and a copy of the modified files should be reported once modifications are
0017 completed, unless these modifications are made through the project's GitHub
0018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
0019 should be developed and included in any deliverable.
0020
0021 2. Redistributions of source code must be done through direct
0022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
0023
0024 3. Redistributions in binary form must reproduce the above copyright notice,
0025 this list of conditions and the following disclaimer in the documentation and/or
0026 other materials provided with the distribution.
0027
0028 4. Usage of the binary form on proprietary applications shall require explicit
0029 prior written permission from the the copyright holders, and due credit should
0030 be given to the copyright holders.
0031
0032 5. Neither the name of the copyright holder nor the names of its contributors
0033 may be used to endorse or promote products derived from this software without
0034 specific prior written permission.
0035
0036 The copyright holders provide no reassurances that the source code provided does
0037 not infringe any patent, copyright, or any other intellectual property rights of
0038 third parties. The copyright holders disclaim any liability to any recipient for
0039 claims brought against recipient by any third party for infringement of that
0040 parties intellectual property rights.
0041
0042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
0043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
0044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
0045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
0046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
0047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
0048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
0049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
0050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
0051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
0052 */
0053
0054 #include <iostream>
0055 #include <ctime>
0056
0057 #include "mtk.h"
0058
0059 void TestDefaultConstructor() {
0060     mtk::Tools::BeginUnitTestNo(1);
0061     mtk::DenseMatrix m1;
0062
0063     mtk::Tools::EndUnitTestNo(1);
0064     mtk::Tools::Assert(m1.data() == nullptr);
0065 }
0066
0067
0068 void TestConstructorWithNumRowsNumCols() {
0069     mtk::Tools::BeginUnitTestNo(2);
0070
0071     int rr = 4;
0072     int cc = 7;
0073
0074     mtk::DenseMatrix m2(rr,cc);
0075
0076     mtk::Tools::EndUnitTestNo(2);
0077
0078     bool assertion =
0079         m2.data() != nullptr && m2.num_rows() == rr && m2.num_cols() ==
0080         cc;
0081
0082     mtk::Tools::Assert(assertion);
0083 }
```

```
00084 }
00085
00086 void TestConstructAsIdentity() {
00087
00088     mtk::Tools::BeginUnitTestNo(3);
00089
00090     int rank = 5;
00091     bool padded = true;
00092     bool transpose = false;
00093
00094     mtk::DenseMatrix m3(rank,padded,transpose);
00095
00096     mtk::DenseMatrix rr(rank + 2,rank);
00097
00098     for (int ii = 0; ii < rank; ++ii) {
00099         rr.SetValue(ii + 1, ii, mtk::kOne);
00100     }
00101
00102     mtk::Tools::EndUnitTestNo(3);
00103     mtk::Tools::Assert(m3 == rr);
00104 }
00105
00106 void TestConstructAsVandermonde() {
00107
00108     mtk::Tools::BeginUnitTestNo(4);
00109
00110     int rank = 5;
00111     bool padded = false;
00112     bool transpose = false;
00113
00114     mtk::DenseMatrix m4(rank,padded,transpose);
00115
00116     mtk::DenseMatrix rr(rank,rank);
00117
00118     for (int ii = 0; ii < rank; ++ii) {
00119         rr.SetValue(ii, ii, mtk::kOne);
00120     }
00121
00122     mtk::Tools::EndUnitTestNo(4);
00123     mtk::Tools::Assert(m4 == rr);
00124 }
00125
00126 void TestSetValueGetValue() {
00127
00128     mtk::Tools::BeginUnitTestNo(5);
00129
00130     int rr = 4;
00131     int cc = 7;
00132
00133     mtk::DenseMatrix m5(rr,cc);
00134
00135     for (auto ii = 0; ii < rr; ++ii) {
00136         for (auto jj = 0; jj < cc; ++jj) {
00137             m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00138         }
00139     }
00140
00141     mtk::Real *vals = m5.data();
00142
00143     bool assertion{true};
00144
00145     for (auto ii = 0; ii < rr && assertion; ++ii) {
00146         for (auto jj = 0; jj < cc && assertion; ++jj) {
00147             assertion = assertion && m5.GetValue(ii,jj) == vals[ii*cc + jj];
00148         }
00149     }
00150
00151     mtk::Tools::EndUnitTestNo(5);
00152     mtk::Tools::Assert(assertion);
00153 }
00154
00155 void TestConstructAsVandermondeTranspose() {
00156
00157     mtk::Tools::BeginUnitTestNo(6);
00158
00159     bool transpose = false;
00160     int generator_length = 3;
00161     int progression_length = 4;
00162
00163     mtk::Real generator[] = {-0.5, 0.5, 1.5};
00164 }
```

```
00165     mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00166     transpose = true;
00167
00168     mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00169     mtk::DenseMatrix rr(progression_length, generator_length);
00170
00171     rr.SetValue(0, 0, 1.0);
00172     rr.SetValue(0, 1, 1.0);
00173     rr.SetValue(0, 2, 1.0);
00174
00175     rr.SetValue(1, 0, -0.5);
00176     rr.SetValue(1, 1, 0.5);
00177     rr.SetValue(1, 2, 1.5);
00178
00179     rr.SetValue(2, 0, 0.25);
00180     rr.SetValue(2, 1, 0.25);
00181     rr.SetValue(2, 2, 2.25);
00182
00183     rr.SetValue(3, 0, -0.125);
00184     rr.SetValue(3, 1, 0.125);
00185     rr.SetValue(3, 2, 3.375);
00186
00187     mtk::Tools::EndUnitTestNo(6);
00188     mtk::Tools::Assert(m7 == rr);
00189 }
00190
00191 void TestKron() {
00192
00193     mtk::Tools::BeginUnitTestNo(7);
00194
00195     bool padded = false;
00196     bool transpose = false;
00197     int lots_of_rows = 2;
00198     int lots_of_cols = 5;
00199
00200     mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00201
00202     mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00203
00204     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00205         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00206             m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00207         }
00208     }
00209
00210     mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00211
00212     mtk::DenseMatrix rr(lots_of_rows*lots_of_rows, lots_of_rows*lots_of_cols);
00213
00214     rr.SetValue(0,0,1.0);
00215     rr.SetValue(0,1,2.0);
00216     rr.SetValue(0,2,3.0);
00217     rr.SetValue(0,3,4.0);
00218     rr.SetValue(0,4,5.0);
00219     rr.SetValue(0,5,0.0);
00220     rr.SetValue(0,6,0.0);
00221     rr.SetValue(0,7,0.0);
00222     rr.SetValue(0,8,0.0);
00223     rr.SetValue(0,9,0.0);
00224
00225     rr.SetValue(1,0,6.0);
00226     rr.SetValue(1,1,7.0);
00227     rr.SetValue(1,2,8.0);
00228     rr.SetValue(1,3,9.0);
00229     rr.SetValue(1,4,10.0);
00230     rr.SetValue(1,5,0.0);
00231     rr.SetValue(1,6,0.0);
00232     rr.SetValue(1,7,0.0);
00233     rr.SetValue(1,8,0.0);
00234     rr.SetValue(1,9,0.0);
00235
00236     rr.SetValue(2,0,0.0);
00237     rr.SetValue(2,1,0.0);
00238     rr.SetValue(2,2,0.0);
00239     rr.SetValue(2,3,0.0);
00240     rr.SetValue(2,4,0.0);
00241     rr.SetValue(2,5,1.0);
00242     rr.SetValue(2,6,2.0);
00243     rr.SetValue(2,7,3.0);
00244     rr.SetValue(2,8,4.0);
00245     rr.SetValue(2,9,5.0);
```

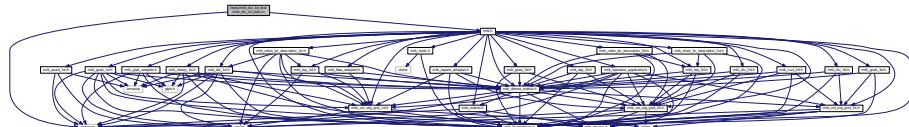
```

00246
00247     rr.SetValue(3,0,0.0);
00248     rr.SetValue(3,1,0.0);
00249     rr.SetValue(3,2,0.0);
00250     rr.SetValue(3,3,0.0);
00251     rr.SetValue(3,4,0.0);
00252     rr.SetValue(3,5,6.0);
00253     rr.SetValue(3,6,7.0);
00254     rr.SetValue(3,7,8.0);
00255     rr.SetValue(3,8,9.0);
00256     rr.SetValue(3,9,10.0);
00257
00258     mtk::Tools::EndUnitTestNo(7);
00259     mtk::Tools::Assert(m10 == rr);
00260 }
00261
00262 void TestConstructWithNumRowsNumColsAssignmentOperator() {
00263
00264     mtk::Tools::BeginUnitTestNo(8);
00265
00266     int lots_of_rows = 4;
00267     int lots_of_cols = 3;
00268     mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00269
00270     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00271         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00272             m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00273         }
00274     }
00275
00276     m11.Transpose();
00277
00278     mtk::DenseMatrix m12;
00279
00280     m12 = m11;
00281
00282     mtk::Tools::EndUnitTestNo(8);
00283     mtk::Tools::Assert(m11 == m12);
00284 }
00285
00286 void TestConstructAsVandermondeTransposeAssignmentOperator()
00287 () {
00288
00289     mtk::Tools::BeginUnitTestNo(9);
00290
00291     bool transpose = false;
00292     int gg_1 = 3;
00293     int progression_length = 4;
00294     mtk::Real gg[] = {-0.5, 0.5, 1.5};
00295
00296     mtk::DenseMatrix m13(gg, gg_1 ,progression_length, transpose);
00297
00298     mtk::DenseMatrix m14;
00299     m14 = m13;
00300
00301     m13.Transpose();
00302
00303     m14 = m13;
00304
00305     mtk::Tools::EndUnitTestNo(9);
00306     mtk::Tools::Assert(m13 == m14);
00307 }
00308
00309 int main () {
00310
00311     std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00312
00313     TestDefaultConstructor();
00314     TestConstructorWithNumRowsNumCols();
00315     TestConstructAsIdentity();
00316     TestConstructAsVandermonde();
00317     TestSetValueGetValue();
00318     TestConstructAsVandermondeTranspose();
00319     TestKron();
00320     TestConstructWithNumRowsNumColsAssignmentOperator();
00321     TestConstructAsVandermondeTransposeAssignmentOperator
00322     ();
00323 }
```

18.139 tests/mtk_div_1d_test/mtk_div_1d_test.cc File Reference

Testing the mimetic 1D divergence, constructed with the CBS algorithm.

```
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_div_1d_test.cc:
```



Functions

- void [TestDefaultConstructorFactoryMethodDefault \(\)](#)
- void [TestDefaultConstructorFactoryMethodFourthOrder \(\)](#)
- void [TestDefaultConstructorFactoryMethodSixthOrder \(\)](#)
- void [TestDefaultConstructorFactoryMethodEightOrderDefThreshold \(\)](#)
- void [TestDefaultConstructorFactoryMethodTenthOrderDefThreshold \(\)](#)
- void [TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold \(\)](#)
- void [TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold \(\)](#)
- void [TestSecondOrderReturnAsDenseMatrixWithGrid \(\)](#)
- void [TestFourthOrderReturnAsDenseMatrixWithGrid \(\)](#)
- void [TestSixthOrderReturnAsDenseMatrixWithGrid \(\)](#)
- int [main \(\)](#)

18.139.1 Detailed Description

Author

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

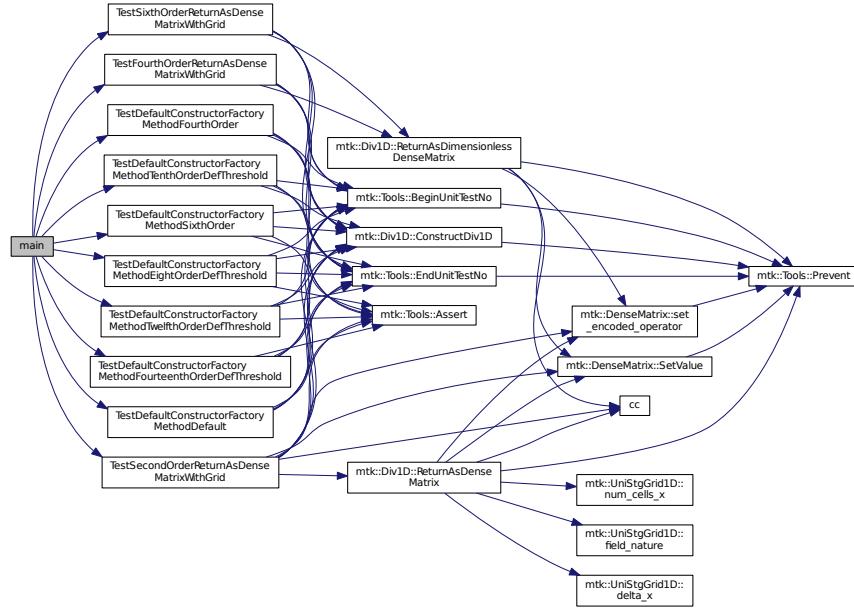
Definition in file [mtk_div_1d_test.cc](#).

18.139.2 Function Documentation

18.139.2.1 int main ()

Definition at line 294 of file [mtk_div_1d_test.cc](#).

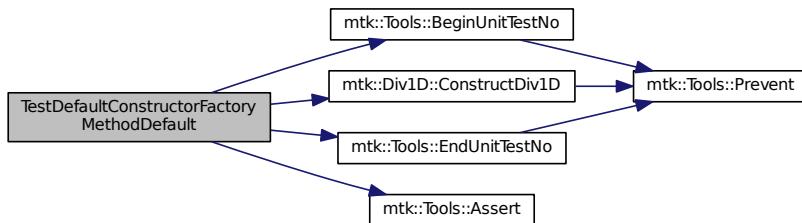
Here is the call graph for this function:



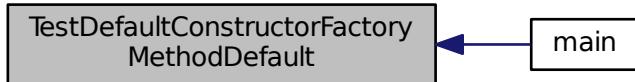
18.139.2.2 void TestDefaultConstructorFactoryMethodDefault()

Definition at line 58 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



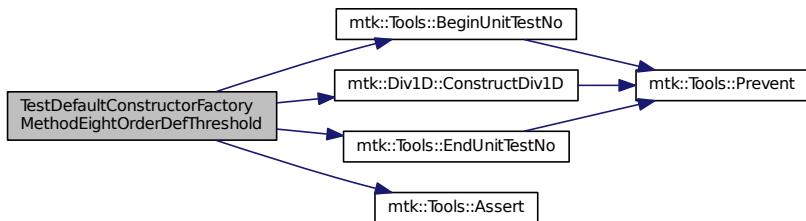
Here is the caller graph for this function:



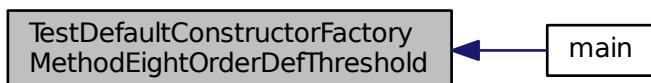
18.139.2.3 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold()

Definition at line 106 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



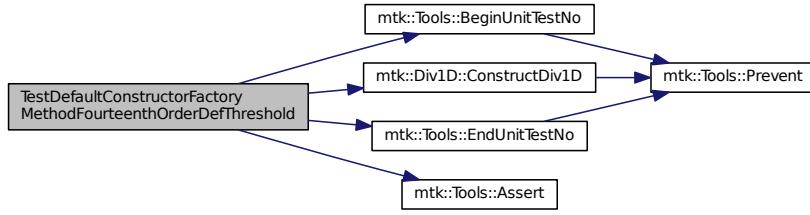
Here is the caller graph for this function:



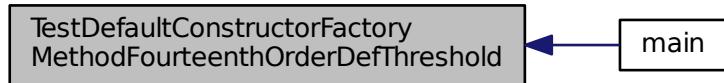
18.139.2.4 void TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold()

Definition at line 154 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



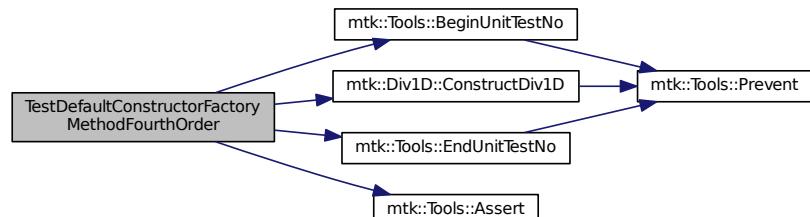
Here is the caller graph for this function:



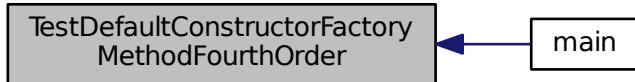
18.139.2.5 void TestDefaultConstructorFactoryMethodFourthOrder()

Definition at line 74 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



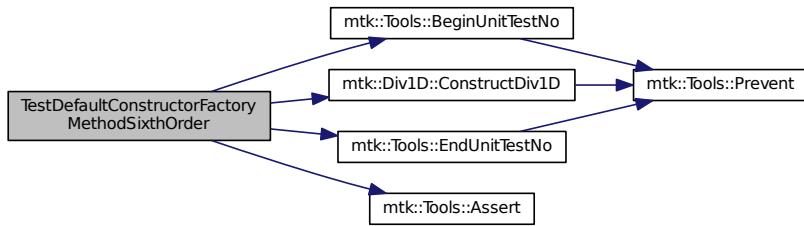
Here is the caller graph for this function:



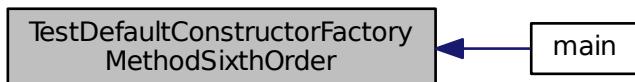
18.139.2.6 void TestDefaultConstructorFactoryMethodSixthOrder()

Definition at line 90 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



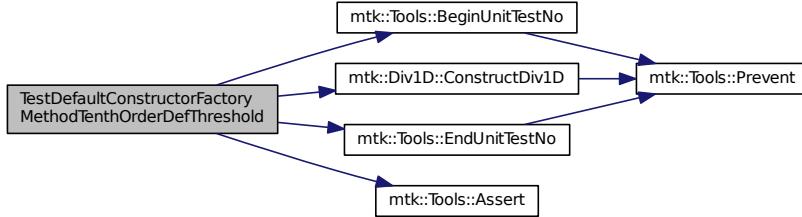
Here is the caller graph for this function:



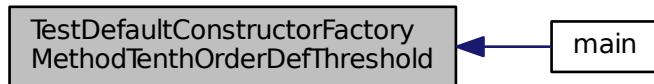
18.139.2.7 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold()

Definition at line 122 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



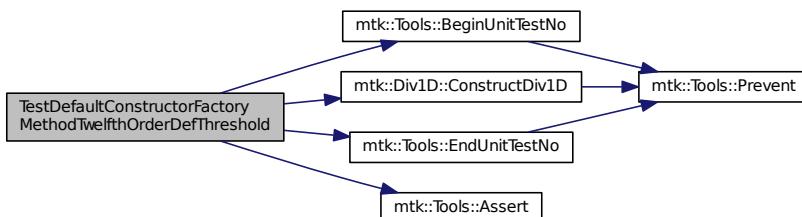
Here is the caller graph for this function:



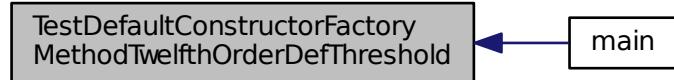
18.139.2.8 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold()

Definition at line 138 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



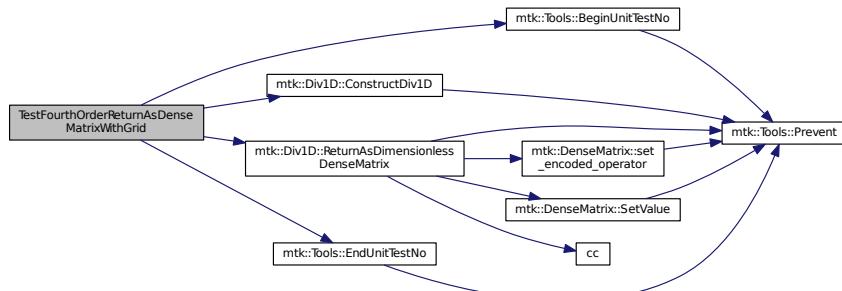
Here is the caller graph for this function:



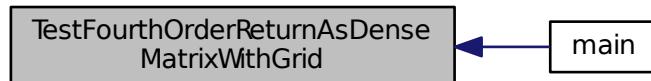
18.139.2.9 void TestFourthOrderReturnAsDenseMatrixWithGrid()

Definition at line 244 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



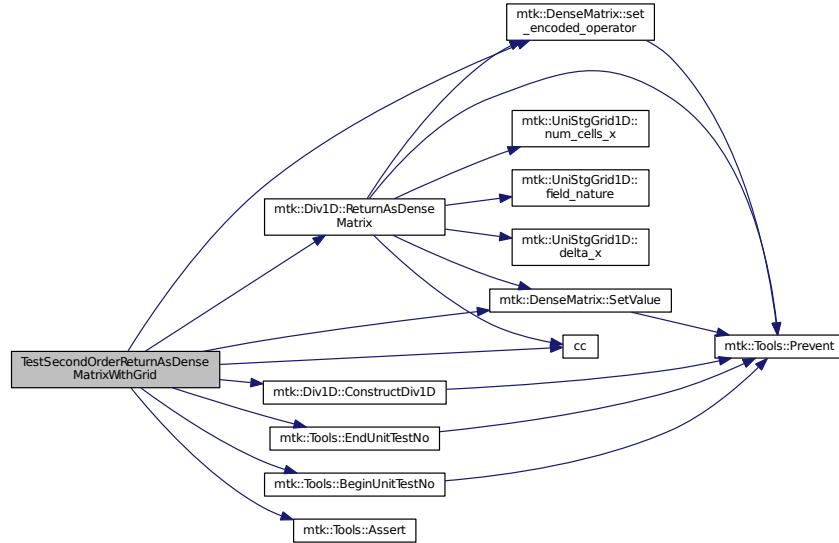
Here is the caller graph for this function:



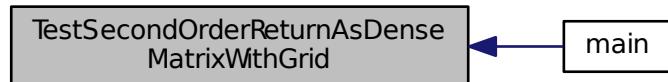
18.139.2.10 void TestSecondOrderReturnAsDenseMatrixWithGrid()

Definition at line 170 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



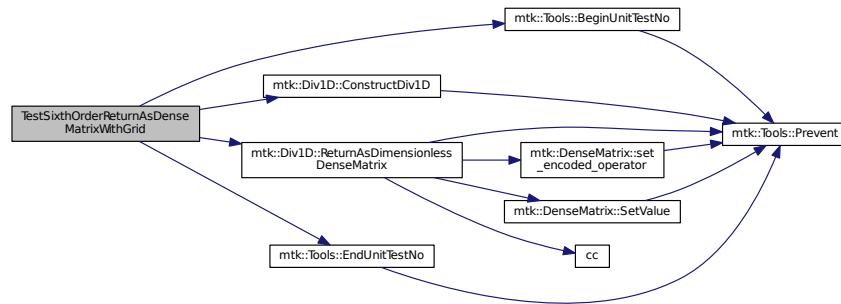
Here is the caller graph for this function:



18.139.2.11 void TestSixthOrderReturnAsDenseMatrixWithGrid()

Definition at line 269 of file [mtk_div_1d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.140 mtk_div_1d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of

```

```
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <iostream>
00055
00056 #include "mtk.h"
00057
00058 void TestDefaultConstructorFactoryMethodDefault() {
00059
00060     mtk::Tools::BeginUnitTestNo(1);
00061
00062     mtk::Div1D div2;
00063
00064     bool assertion = div2.ConstructDiv1D();
00065
00066     if (!assertion) {
00067         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00068     }
00069
00070     mtk::Tools::EndUnitTestNo(1);
00071     mtk::Tools::Assert(assertion);
00072 }
00073
00074 void TestDefaultConstructorFactoryMethodFourthOrder() {
00075
00076     mtk::Tools::BeginUnitTestNo(2);
00077
00078     mtk::Div1D div4;
00079
00080     bool assertion = div4.ConstructDiv1D(4);
00081
00082     if (!assertion) {
00083         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00084     }
00085
00086     mtk::Tools::EndUnitTestNo(2);
00087     mtk::Tools::Assert(assertion);
00088 }
00089
00090 void TestDefaultConstructorFactoryMethodSixthOrder() {
00091
00092     mtk::Tools::BeginUnitTestNo(3);
00093
00094     mtk::Div1D div6;
00095
00096     bool assertion = div6.ConstructDiv1D(6);
00097
00098     if (!assertion) {
00099         std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00100     }
00101
00102     mtk::Tools::EndUnitTestNo(3);
00103     mtk::Tools::Assert(assertion);
00104 }
00105
00106 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00107     () {
00108
00109     mtk::Tools::BeginUnitTestNo(4);
00110
00111     mtk::Div1D div8;
00112
00113     bool assertion = div8.ConstructDiv1D(8);
00114
00115     if (!assertion) {
00116         std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00117     }
00118 }
```

```
00118     mtk::Tools::EndUnitTestNo(4);
00119     mtk::Tools::Assert(assertion);
00120 }
00121
00122 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00123 () {
00124     mtk::Tools::BeginUnitTestNo(5);
00125
00126     mtk::Div1D div10;
00127
00128     bool assertion = div10.ConstructDiv1D(10);
00129
00130     if (!assertion) {
00131         std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00132     }
00133
00134     mtk::Tools::EndUnitTestNo(5);
00135     mtk::Tools::Assert(assertion);
00136 }
00137
00138 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold
00139 () {
00140     mtk::Tools::BeginUnitTestNo(6);
00141
00142     mtk::Div1D div12;
00143
00144     bool assertion = div12.ConstructDiv1D(12);
00145
00146     if (!assertion) {
00147         std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00148     }
00149
00150     mtk::Tools::EndUnitTestNo(6);
00151     mtk::Tools::Assert(assertion);
00152 }
00153
00154 void TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold
00155 () {
00156     mtk::Tools::BeginUnitTestNo(7);
00157
00158     mtk::Div1D div14;
00159
00160     bool assertion = div14.ConstructDiv1D(14);
00161
00162     if (!assertion) {
00163         std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00164     }
00165
00166     mtk::Tools::EndUnitTestNo(7);
00167     mtk::Tools::Assert(assertion);
00168 }
00169
00170 void TestSecondOrderReturnAsDenseMatrixWithGrid() {
00171
00172     mtk::Tools::BeginUnitTestNo(8);
00173
00174     mtk::Div1D div2;
00175
00176     bool assertion = div2.ConstructDiv1D();
00177
00178     if (!assertion) {
00179         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00180     }
00181
00182     mtk::UniStgGrid1D grid(0.0, 1.0, 5, mtk::FieldNature::VECTOR);
00183
00184     mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00185
00186     int rr{7};
00187     int cc{6};
00188
00189     mtk::DenseMatrix ref(rr, cc);
00190
00191     ref.set_encoded_operator(mtk::EncodedOperator::DIVERGENCE
00192 );
00193
00194 // Row 2.
00194     ref.SetValue(1, 0, -5.0);
```

```

00195     ref.SetValue(1,1,5.0);
00196     ref.SetValue(1,2,0.0);
00197     ref.SetValue(1,3,0.0);
00198     ref.SetValue(1,4,0.0);
00199     ref.SetValue(1,5,0.0);
00200     ref.SetValue(1,6,0.0);
00201
00202     // Row 3.
00203     ref.SetValue(2,0,0.0);
00204     ref.SetValue(2,1,-5.0);
00205     ref.SetValue(2,2,5.0);
00206     ref.SetValue(2,3,0.0);
00207     ref.SetValue(2,4,0.0);
00208     ref.SetValue(2,5,0.0);
00209     ref.SetValue(2,6,0.0);
00210
00211     // Row 4.
00212     ref.SetValue(3,0,0.0);
00213     ref.SetValue(3,1,0.0);
00214     ref.SetValue(3,2,-5.0);
00215     ref.SetValue(3,3,5.0);
00216     ref.SetValue(3,4,0.0);
00217     ref.SetValue(3,5,0.0);
00218     ref.SetValue(3,6,0.0);
00219
00220     // Row 5.
00221     ref.SetValue(4,0,0.0);
00222     ref.SetValue(4,1,0.0);
00223     ref.SetValue(4,2,0.0);
00224     ref.SetValue(4,3,-5.0);
00225     ref.SetValue(4,4,5.0);
00226     ref.SetValue(4,5,0.0);
00227     ref.SetValue(4,6,0.0);
00228
00229     // Row 6.
00230     ref.SetValue(5,0,0.0);
00231     ref.SetValue(5,1,0.0);
00232     ref.SetValue(5,2,0.0);
00233     ref.SetValue(5,3,0.0);
00234     ref.SetValue(5,4,-5.0);
00235     ref.SetValue(5,5,5.0);
00236     ref.SetValue(5,6,0.0);
00237
00238     assertion = assertion && (div2m == ref);
00239
00240     mtk::Tools::EndUnitTestNo(8);
00241     mtk::Tools::Assert(assertion);
00242 }
00243
00244 void TestFourthOrderReturnAsDenseMatrixWithGrid() {
00245
00246     mtk::Tools::BeginUnitTestNo(9);
00247
00248     mtk::Div1D div4;
00249
00250     bool assertion = div4.ConstructDiv1D(4);
00251
00252     if (!assertion) {
00253         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00254     }
00255
00256     std::cout << div4 << std::endl;
00257
00258     mtk::UniStgGrid1D grid(0.0, 1.0, 12, mtk::FieldNature::VECTOR);
00259
00260     std::cout << grid << std::endl;
00261
00262     mtk::DenseMatrix div4m(div4.ReturnAsDimensionlessDenseMatrix
00263     (12));
00264
00265     std::cout << div4m << std::endl;
00266
00267     mtk::Tools::EndUnitTestNo(9);
00268 }
00269
00270 void TestSixthOrderReturnAsDenseMatrixWithGrid() {
00271
00272     mtk::Tools::BeginUnitTestNo(10);
00273
00274     mtk::Div1D div6;

```

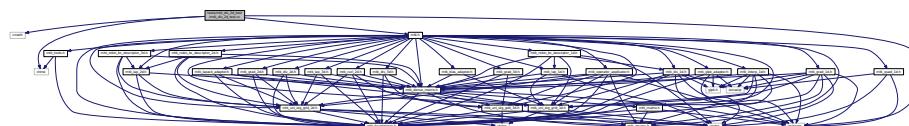
```

00275     bool assertion = div6.ConstructDiv1D(6);
00276
00277     if (!assertion) {
00278         std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00279     }
00280
00281     std::cout << div6 << std::endl;
00282
00283     mtk::UniStgGrid1D grid(0.0, 1.0, 20, mtk::FieldNature::VECTOR);
00284
00285     std::cout << grid << std::endl;
00286
00287     mtk::DenseMatrix div6m(div6.ReturnAsDimensionlessDenseMatrix
00288     (20));
00289
00290     std::cout << div6m << std::endl;
00291
00292     mtk::Tools::EndUnitTestNo(10);
00293 }
00294
00295 int main () {
00296     std::cout << "Testing mtk::Div1D class." << std::endl;
00297
00298     TestDefaultConstructorFactoryMethodDefault();
00299     TestDefaultConstructorFactoryMethodFourthOrder();
00300     TestDefaultConstructorFactoryMethodSixthOrder();
00301     TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00302     ();
00303     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00304     ();
00305     TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold
00306     ();
00307     TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold
00308     ();
00309
00310     TestSecondOrderReturnAsDenseMatrixWithGrid();
00311     TestFourthOrderReturnAsDenseMatrixWithGrid();
00312     TestSixthOrderReturnAsDenseMatrixWithGrid();
00313 }
```

18.141 tests/mtk_div_2d_test/mtk_div_2d_test.cc File Reference

Test file for the [mtk::Div2D](#) class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_div_2d_test.cc:
```



Functions

- void [TestDefaultConstructorFactory](#) ()
- void [TestReturnAsDenseMatrixWriteToFile](#) ()
- int [main](#) ()

18.141.1 Detailed Description

Author

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

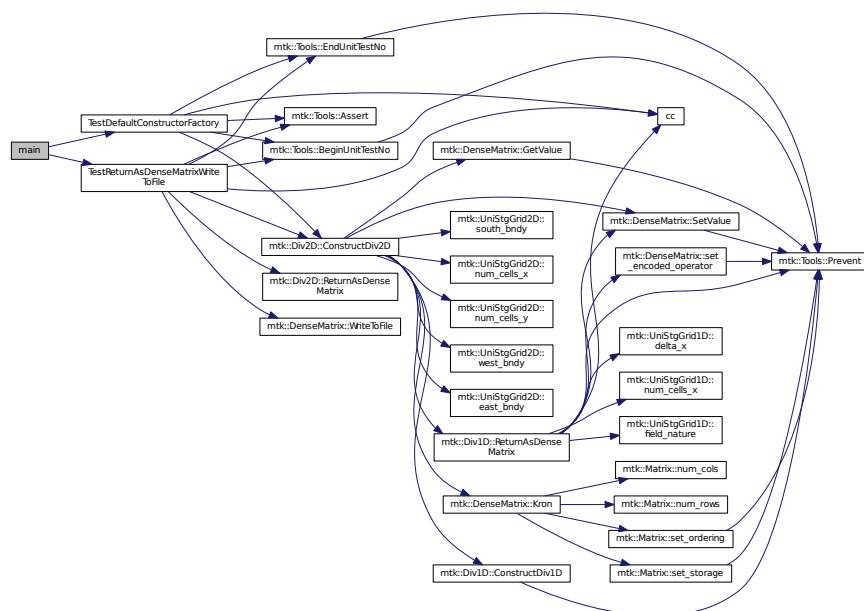
Definition in file [mtk_div_2d_test.cc](#).

18.141.2 Function Documentation

18.141.2.1 int main ()

Definition at line 125 of file [mtk_div_2d_test.cc](#).

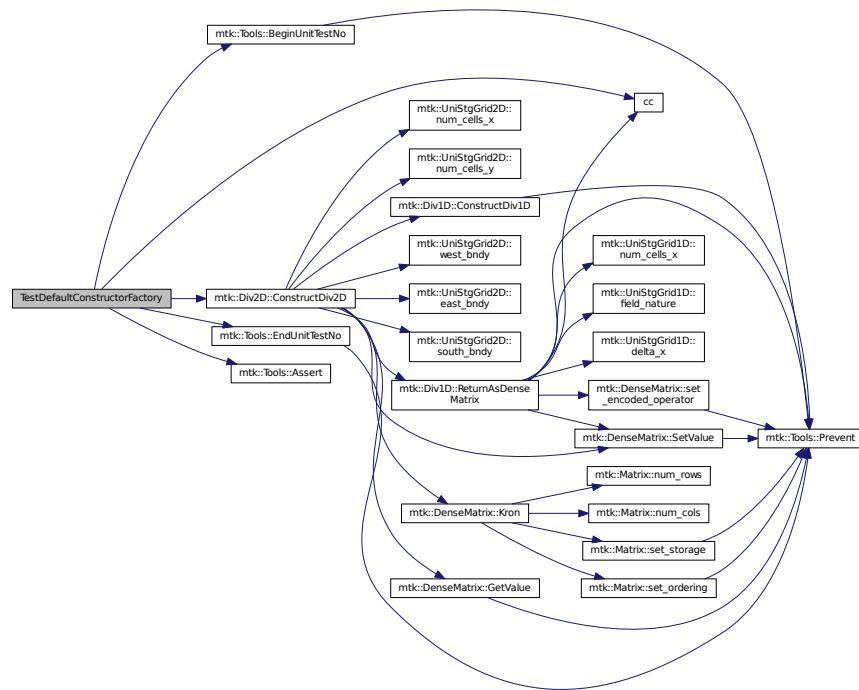
Here is the call graph for this function:



18.141.2.2 void TestDefaultConstructorFactory ()

Definition at line 61 of file [mtk_div_2d_test.cc](#).

Here is the call graph for this function:



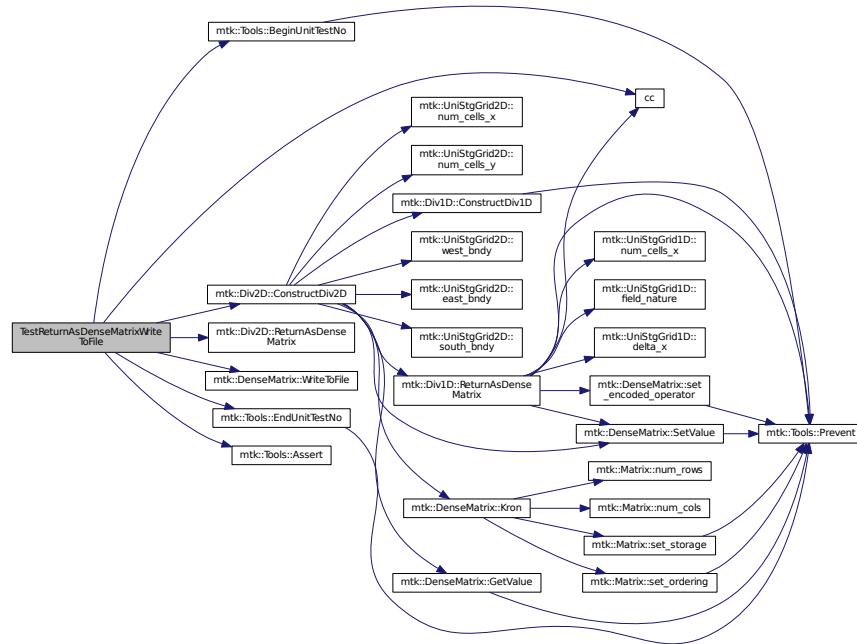
Here is the caller graph for this function:



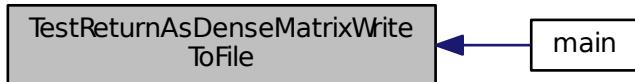
18.141.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 87 of file [mtk_div_2d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.142 mtk_div_2d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk

```

```
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Div2D dd;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real ee = 1.0;
00071
00072     int nn = 5;
00073     int mm = 5;
00074
00075     mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00076
00077     bool assertion = dd.ConstructDiv2D(ddg);
00078
00079     if (!assertion) {
00080         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00081     }
00082
00083     mtk::Tools::EndUnitTestNo(1);
00084     mtk::Tools::Assert(assertion);
00085 }
00086
00087 void TestReturnAsDenseMatrixWriteToFile() {
00088
00089     mtk::Tools::BeginUnitTestNo(2);
00090
00091     mtk::Div2D dd;
00092
00093     mtk::Real aa = 0.0;
00094     mtk::Real bb = 1.0;
00095     mtk::Real cc = 0.0;
00096     mtk::Real ee = 1.0;
00097
00098     int nn = 5;
00099     int mm = 5;
00100
00101     mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00102
00103     bool assertion = dd.ConstructDiv2D(ddg);
```

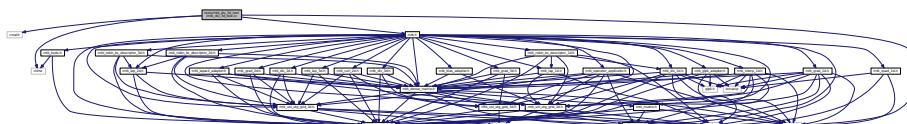
```

00104
00105     if (!assertion) {
00106         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00107     }
00108
00109     mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00110
00111     assertion = assertion && (ddm.num_rows() != mtk::kZero);
00112
00113     std::cout << ddm << std::endl;
00114
00115     assertion = assertion && ddm.WriteToFile("mtk_div_2d_test_02_ddm.dat");
00116
00117     if (!assertion) {
00118         std::cerr << "Error writing to file." << std::endl;
00119     }
00120
00121     mtk::Tools::EndUnitTestNo(2);
00122     mtk::Tools::Assert(assertion);
00123 }
00124
00125 int main () {
00126
00127     std::cout << "Testing mtk::Div2D class." << std::endl;
00128
00129     TestDefaultConstructorFactory();
00130     TestReturnAsDenseMatrixWriteToFile();
00131 }
```

18.143 tests/mtk_div_3d_test/mtk_div_3d_test.cc File Reference

Test file for the `mtk::Div3D` class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_div_3d_test.cc:
```



Functions

- void `TestDefaultConstructorFactory ()`
- void `TestReturnAsDenseMatrixWriteToFile ()`
- int `main ()`

18.143.1 Detailed Description

Author

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

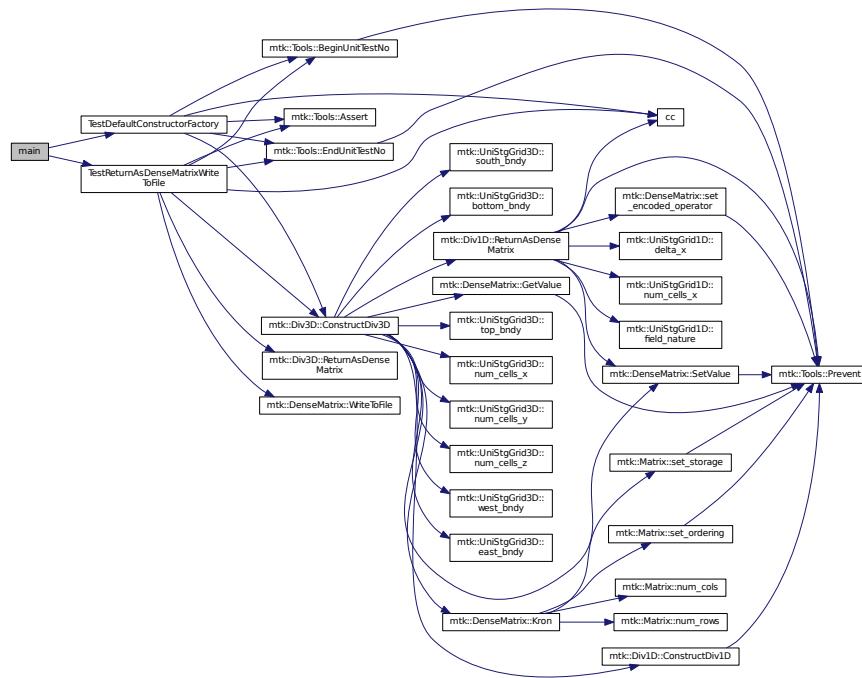
Definition in file [mtk_div_3d_test.cc](#).

18.143.2 Function Documentation

18.143.2.1 int main ()

Definition at line 131 of file [mtk_div_3d_test.cc](#).

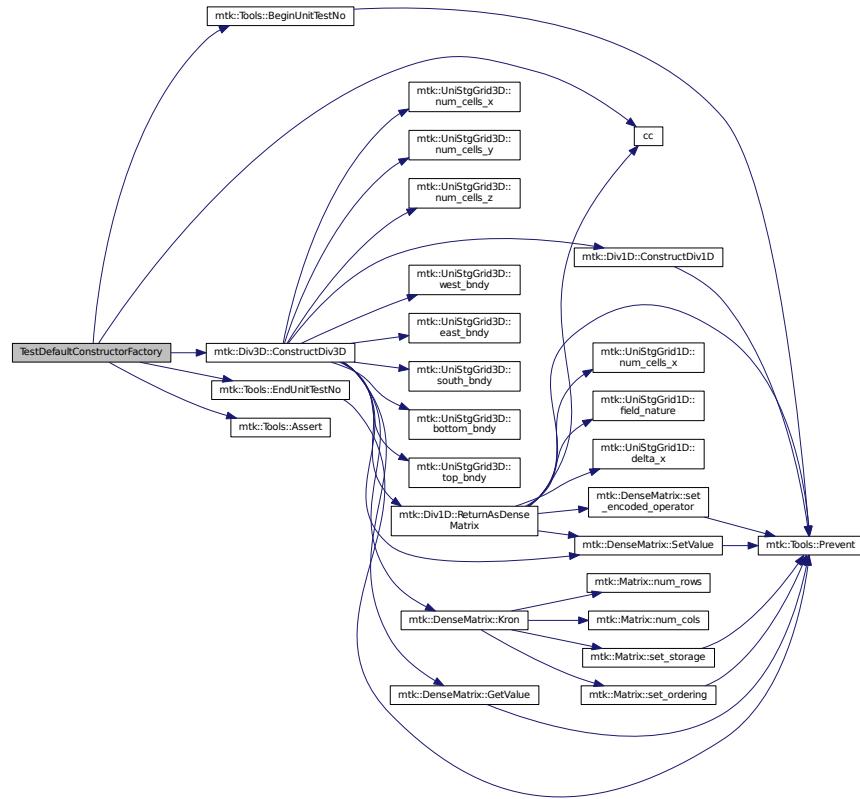
Here is the call graph for this function:



18.143.2.2 void TestDefaultConstructorFactory()

Definition at line 61 of file [mtk/div_3d/test.cc](#).

Here is the call graph for this function:



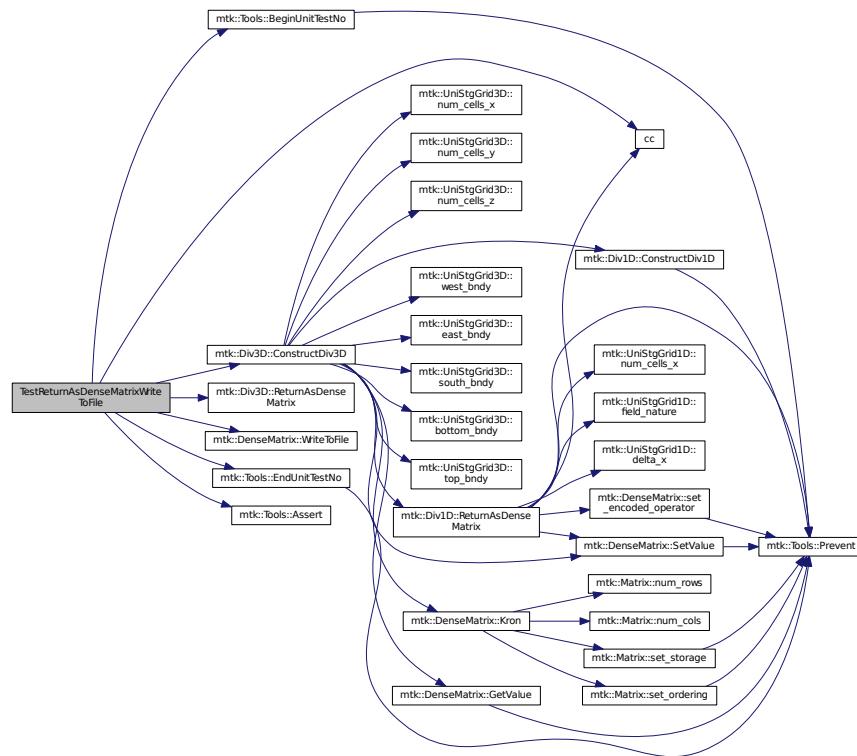
Here is the caller graph for this function:



18.143.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 90 of file [mtk_div_3d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.144 mtk_div_3d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are

```

```

00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Div3D div;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real dd = 1.0;
00071     mtk::Real ee = 0.0;
00072     mtk::Real ff = 1.0;
00073
00074     int nn = 5;
00075     int mm = 5;
00076     int oo = 5;
00077
00078     mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00079
00080     bool assertion = div.ConstructDiv3D(divg);
00081
00082     if (!assertion) {
00083         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00084     }
00085
00086     mtk::Tools::EndUnitTestNo(1);
00087     mtk::Tools::Assert(assertion);
00088 }
00089
00090 void TestReturnAsDenseMatrixWriteToFile() {
00091
00092     mtk::Tools::BeginUnitTestNo(2);
00093
00094     mtk::Div3D div;
00095
00096     mtk::Real aa = 0.0;
00097     mtk::Real bb = 1.0;

```

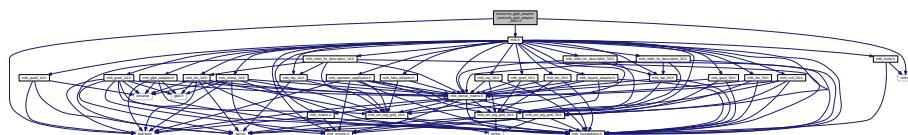
```

00098     mtk::Real cc = 0.0;
00099     mtk::Real dd = 1.0;
00100     mtk::Real ee = 0.0;
00101     mtk::Real ff = 1.0;
00102
00103     int nn = 5;
00104     int mm = 5;
00105     int oo = 5;
00106
00107     mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00108
00109     bool assertion = div.ConstructDiv3D(divg);
00110
00111     if (!assertion) {
00112         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00113     }
00114
00115     mtk::DenseMatrix divm(div.ReturnAsDenseMatrix());
00116
00117     assertion = assertion && (divm.num_rows() != mtk::kZero);
00118
00119     std::cout << divm << std::endl;
00120
00121     assertion = assertion && divm.WriteToFile("mtk_div_3d_test_02_divm.dat");
00122
00123     if (!assertion) {
00124         std::cerr << "Error writing to file." << std::endl;
00125     }
00126
00127     mtk::Tools::EndUnitTestNo(2);
00128     mtk::Tools::Assert(assertion);
00129 }
00130
00131 int main () {
00132
00133     std::cout << "Testing mtk::Div3D class." << std::endl;
00134
00135     TestDefaultConstructorFactory();
00136     TestReturnAsDenseMatrixWriteToFile();
00137 }
```

18.145 tests/mtk_glpk_adapter_test/mtk_glpk_adapter_test.cc File Reference

Test file for the `mtk::GLPKAdapter` class.

```
#include <iostream>
#include <ctime>
#include "mtk.h"
Include dependency graph for mtk_glpk_adapter_test.cc:
```



Functions

- void `Test1 ()`
- int `main ()`

18.145.1 Detailed Description

Author

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

Todo Test the `mtk::GLPKAdapter` class.

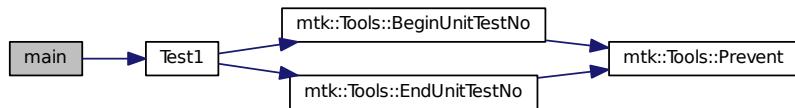
Definition in file [mtk_glpk_adapter_test.cc](#).

18.145.2 Function Documentation

18.145.2.1 int main()

Definition at line 68 of file [mtk_glpk_adapter_test.cc](#).

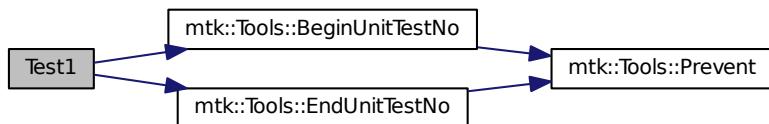
Here is the call graph for this function:



18.145.2.2 void Test1()

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

Here is the call graph for this function:



Here is the caller graph for this function:



18.146 mtk_glpk_adapter_test.cc

```

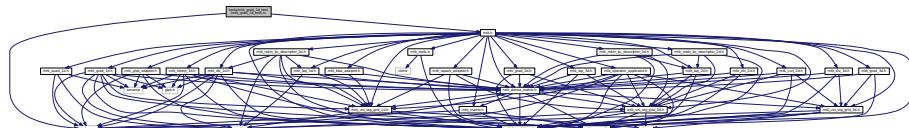
00001 /*
00010 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void Test1() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Tools::EndUnitTestNo(1);
00066 }
00067
00068 int main () {
00069
00070     std::cout << "Testing mtk::GLPKAdapter class." << std::endl;
00071
00072     Test1();
00073 }
```

18.147 tests/mtk_grad_1d_test/mtk_grad_1d_test.cc File Reference

Testing the mimetic 1D gradient, constructed with the CBS algorithm.

```
#include <iostream>
#include "mtk.h"
```

Include dependency graph for mtk_grad_1d_test.cc:



Functions

- void [TestDefaultConstructorFactoryMethodDefault \(\)](#)
- void [TestDefaultConstructorFactoryMethodFourthOrder \(\)](#)
- void [TestDefaultConstructorFactoryMethodSixthOrder \(\)](#)
- void [TestDefaultConstructorFactoryMethodEightOrderDefThreshold \(\)](#)
- void [TestDefaultConstructorFactoryMethodTenthOrderDefThreshold \(\)](#)
- void [TestReturnAsDenseMatrixWithGrid \(\)](#)
- void [TestReturnAsDimensionlessDenseMatrix \(\)](#)
- void [TestWriteToFile \(\)](#)
- void [TestMimBndy \(\)](#)
- int [main \(\)](#)

18.147.1 Detailed Description

Author

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

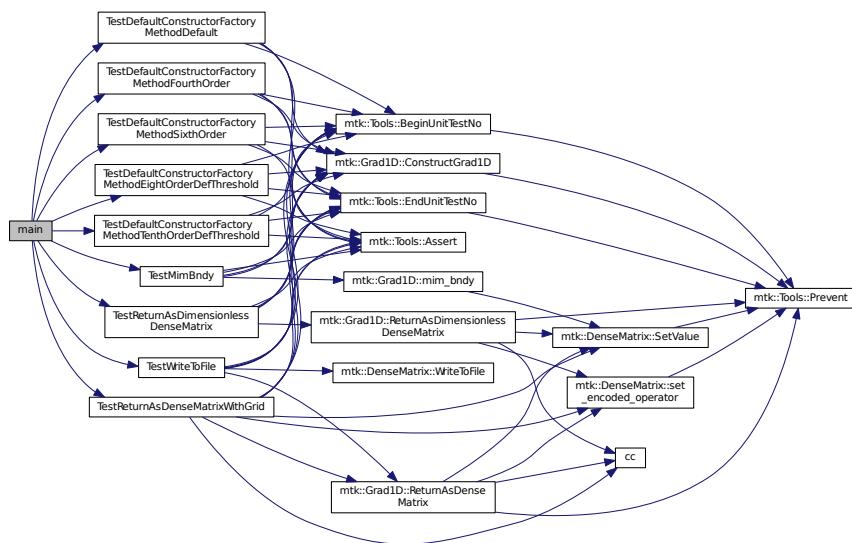
Definition in file [mtk_grad_1d_test.cc](#).

18.147.2 Function Documentation

18.147.2.1 int main ()

Definition at line 303 of file [mtk_grad_1d_test.cc](#).

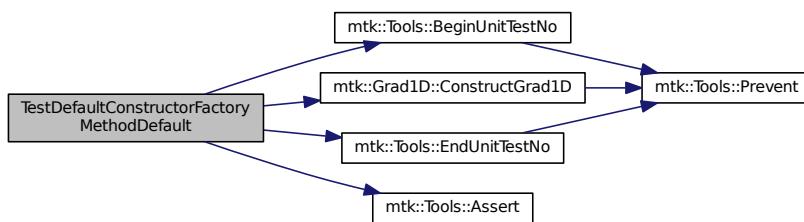
Here is the call graph for this function:



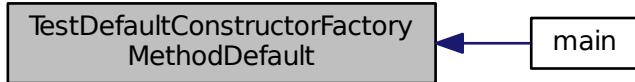
18.147.2.2 void TestDefaultConstructorFactoryMethodDefault ()

Definition at line 58 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



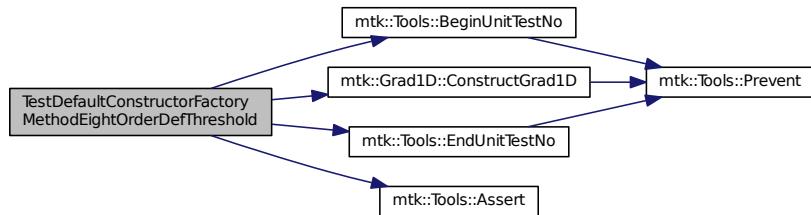
Here is the caller graph for this function:



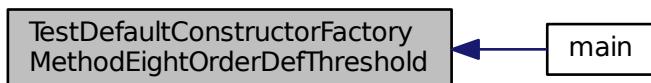
18.147.2.3 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold()

Definition at line 113 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



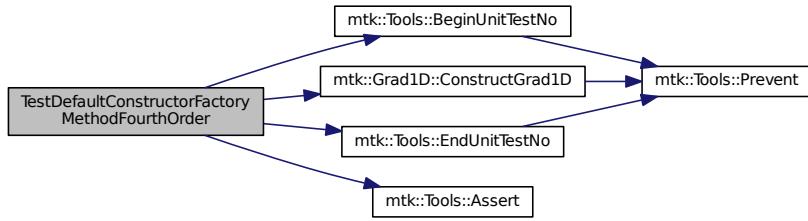
Here is the caller graph for this function:



18.147.2.4 void TestDefaultConstructorFactoryMethodFourthOrder()

Definition at line 77 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



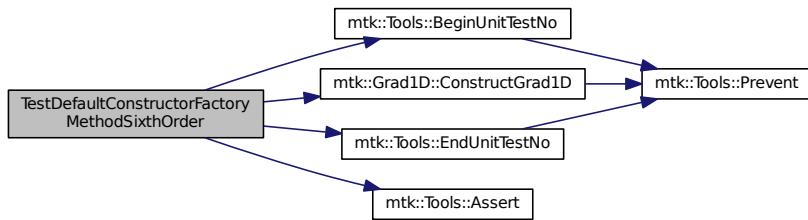
Here is the caller graph for this function:



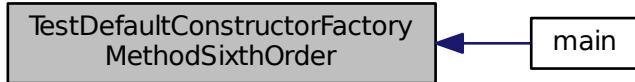
18.147.2.5 void TestDefaultConstructorFactoryMethodSixthOrder()

Definition at line 95 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



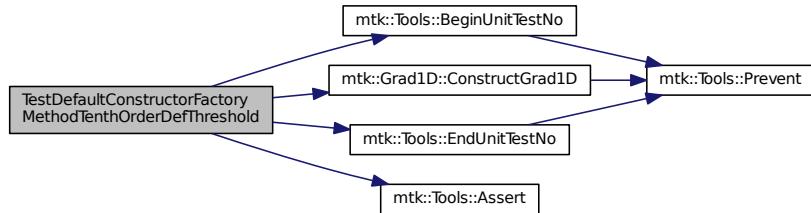
Here is the caller graph for this function:



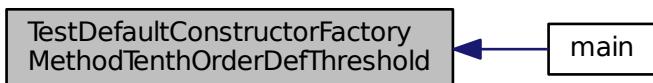
18.147.2.6 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold ()

Definition at line 131 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



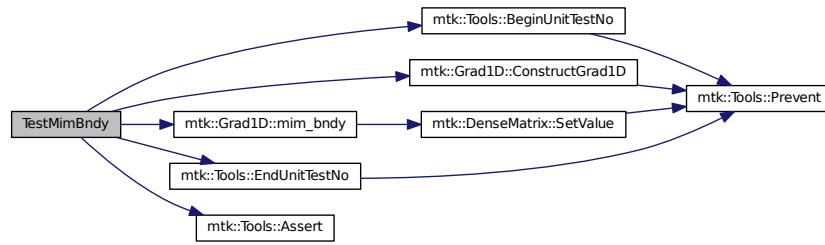
Here is the caller graph for this function:



18.147.2.7 void TestMimBndy ()

Definition at line 281 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



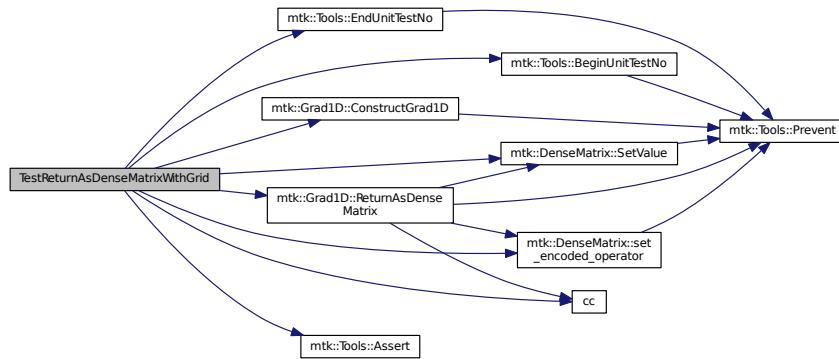
Here is the caller graph for this function:



18.147.2.8 void TestReturnAsDenseMatrixWithGrid()

Definition at line 149 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



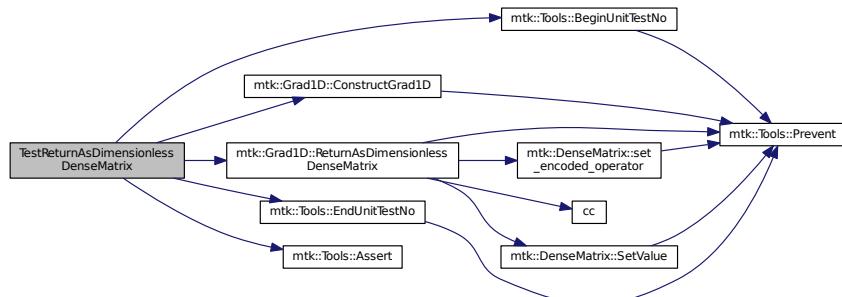
Here is the caller graph for this function:



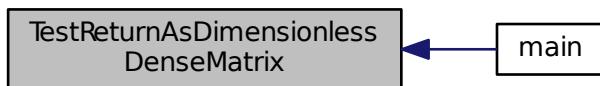
18.147.2.9 void TestReturnAsDimensionlessDenseMatrix ()

Definition at line 233 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



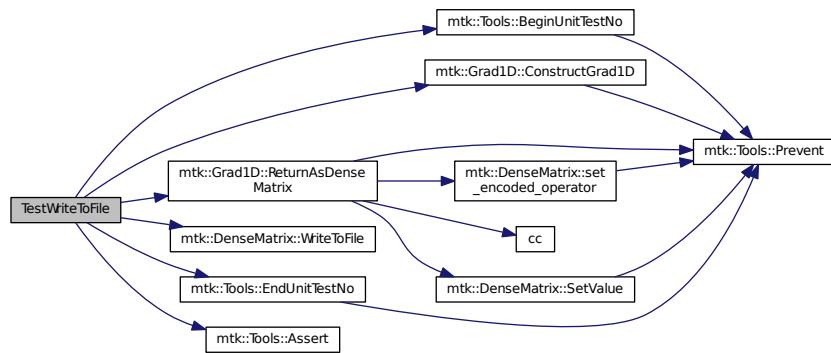
Here is the caller graph for this function:



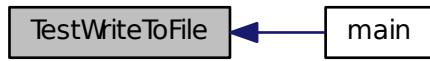
18.147.2.10 void TestWriteToFile ()

Definition at line 253 of file [mtk_grad_1d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.148 mtk_grad_1d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does

```

```
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <iostream>
00055
00056 #include "mtk.h"
00057
00058 void TestDefaultConstructorFactoryMethodDefault() {
00059     mtk::Tools::BeginUnitTestNo(1);
00060
00061     mtk::Grad1D grad2;
00062
00063     bool assertion = grad2.ConstructGrad1D();
00064
00065     if (!assertion) {
00066         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00067     }
00068
00069 }
00070
00071     std::cout << grad2 << std::endl;
00072
00073     mtk::Tools::EndUnitTestNo(1);
00074     mtk::Tools::Assert(assertion);
00075 }
00076
00077 void TestDefaultConstructorFactoryMethodFourthOrder() {
00078
00079     mtk::Tools::BeginUnitTestNo(2);
00080
00081     mtk::Grad1D grad4;
00082
00083     bool assertion = grad4.ConstructGrad1D(4);
00084
00085     if (!assertion) {
00086         std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00087     }
00088
00089     std::cout << grad4 << std::endl;
00090
00091     mtk::Tools::EndUnitTestNo(2);
00092     mtk::Tools::Assert(assertion);
00093 }
00094
00095 void TestDefaultConstructorFactoryMethodSixthOrder() {
00096
00097     mtk::Tools::BeginUnitTestNo(3);
00098
00099     mtk::Grad1D grad6;
00100
00101     bool assertion = grad6.ConstructGrad1D(6);
00102
00103     if (!assertion) {
00104         std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00105     }
00106
00107     std::cout << grad6 << std::endl;
00108
00109     mtk::Tools::EndUnitTestNo(3);
00110     mtk::Tools::Assert(assertion);
00111 }
00112
00113 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00114     () {
00115
00116     mtk::Tools::BeginUnitTestNo(4);
```

```
00117     mtk::Grad1D grad8;
00118
00119     bool assertion = grad8.ConstructGrad1D(8);
00120
00121     if (!assertion) {
00122         std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00123     }
00124
00125     std::cout << grad8 << std::endl;
00126
00127     mtk::Tools::EndUnitTestNo(4);
00128     mtk::Tools::Assert(assertion);
00129 }
00130
00131 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00132 () {
00133     mtk::Tools::BeginUnitTestNo(5);
00134
00135     mtk::Grad1D grad10;
00136
00137     bool assertion = grad10.ConstructGrad1D(10);
00138
00139     if (!assertion) {
00140         std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00141     }
00142
00143     std::cout << grad10 << std::endl;
00144
00145     mtk::Tools::EndUnitTestNo(5);
00146     mtk::Tools::Assert(assertion);
00147 }
00148
00149 void TestReturnAsDenseMatrixWithGrid() {
00150
00151     mtk::Tools::BeginUnitTestNo(6);
00152
00153     mtk::Grad1D grad2;
00154
00155     bool assertion = grad2.ConstructGrad1D();
00156
00157     if (!assertion) {
00158         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00159     }
00160
00161     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00162
00163     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00164
00165     int rr{6};
00166     int cc{7};
00167
00168     mtk::DenseMatrix ref(rr, cc);
00169
00170     ref.set_encoded_operator(mtk::EncodedOperator::GRADIENT
00171 );
00172
00173 // Row 1.
00174 ref.SetValue(0,0,-13.3333);
00175 ref.SetValue(0,1,15);
00176 ref.SetValue(0,2,-1.66667);
00177 ref.SetValue(0,3,0.0);
00178 ref.SetValue(0,4,0.0);
00179 ref.SetValue(0,5,0.0);
00180 ref.SetValue(0,6,0.0);
00181
00182 // Row 2.
00183 ref.SetValue(1,0,0.0);
00184 ref.SetValue(1,1,-5.0);
00185 ref.SetValue(1,2,5.0);
00186 ref.SetValue(1,3,0.0);
00187 ref.SetValue(1,4,0.0);
00188 ref.SetValue(1,5,0.0);
00189 ref.SetValue(1,6,0.0);
00190
00191 // Row 3.
00192 ref.SetValue(2,0,0.0);
00193 ref.SetValue(2,1,0.0);
00194 ref.SetValue(2,2,-5.0);
00195 ref.SetValue(2,3,5.0);
00196 ref.SetValue(2,4,0.0);
```

```

00196     ref.SetValue(2,5,0.0);
00197     ref.SetValue(2,6,0.0);
00198
00199 // Row 4.
00200 ref.SetValue(3,0,0.0);
00201 ref.SetValue(3,1,0.0);
00202 ref.SetValue(3,2,0.0);
00203 ref.SetValue(3,3,-5.0);
00204 ref.SetValue(3,4,5.0);
00205 ref.SetValue(3,5,0.0);
00206 ref.SetValue(3,6,0.0);
00207
00208 // Row 5.
00209 ref.SetValue(4,0,0.0);
00210 ref.SetValue(4,1,0.0);
00211 ref.SetValue(4,2,0.0);
00212 ref.SetValue(4,3,0.0);
00213 ref.SetValue(4,4,-5.0);
00214 ref.SetValue(4,5,5.0);
00215 ref.SetValue(4,6,0.0);
00216
00217 // Row 6.
00218 ref.SetValue(5,0,0.0);
00219 ref.SetValue(5,1,0.0);
00220 ref.SetValue(5,2,0.0);
00221 ref.SetValue(5,3,0.0);
00222 ref.SetValue(5,4,1.66667);
00223 ref.SetValue(5,5,-15.0);
00224 ref.SetValue(5,6,13.3333);
00225
00226 std::cout << grad2m << std::endl;
00227 std::cout << ref << std::endl;
00228
00229 mtk::Tools::EndUnitTestNo(6);
00230 mtk::Tools::Assert(grad2m == ref);
00231 }
00232
00233 void TestReturnAsDimensionlessDenseMatrix() {
00234
00235     mtk::Tools::BeginUnitTestNo(7);
00236
00237     mtk::Grad1D grad4;
00238
00239     bool assertion = grad4.ConstructGrad1D(4);
00240
00241     if (!assertion) {
00242         std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00243     }
00244
00245     mtk::DenseMatrix grad4m(grad4.ReturnAsDimensionlessDenseMatrix
00246     (10));
00247
00248     std::cout << grad4m << std::endl;
00249
00250     mtk::Tools::EndUnitTestNo(7);
00251     mtk::Tools::Assert(assertion);
00252 }
00253
00254 void TestWriteToFile() {
00255
00256     mtk::Tools::BeginUnitTestNo(8);
00257
00258     mtk::Grad1D grad2;
00259
00260     bool assertion = grad2.ConstructGrad1D();
00261
00262     if (!assertion) {
00263         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00264     }
00265
00266     mtk::UniStgGrid1D grid(0.0, 1.0, 50);
00267
00268     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00269
00270     std::cout << grad2m << std::endl;
00271
00272     assertion = assertion && grad2m.WriteLine("mtk_grad_1d_test_08_grad2m.dat");
00273
00274     if (!assertion) {
00275         std::cerr << "Error writing to file." << std::endl;
00276     }

```

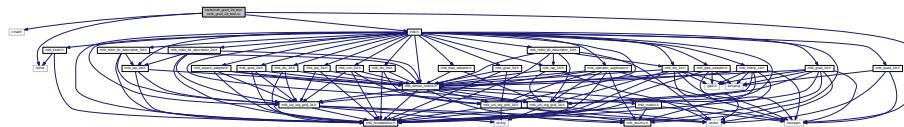
```

00276
00277     mtk::Tools::EndUnitTestNo(8);
00278     mtk::Tools::Assert(assertion);
00279 }
00280
00281 void TestMimBndy() {
00282
00283     mtk::Tools::BeginUnitTestNo(9);
00284
00285     mtk::Grad1D grad2;
00286
00287     bool assertion = grad2.ConstructGrad1D();
00288
00289     if (!assertion) {
00290         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00291     }
00292
00293     std::cout << grad2 << std::endl;
00294
00295     mtk::DenseMatrix grad2m(grad2.mim_bndy());
00296
00297     std::cout << grad2m << std::endl;
00298
00299     mtk::Tools::EndUnitTestNo(9);
00300     mtk::Tools::Assert(assertion);
00301 }
00302
00303 int main () {
00304
00305     std::cout << "Testing mtk::Grad1D class." << std::endl;
00306
00307     TestDefaultConstructorFactoryMethodDefault();
00308     TestDefaultConstructorFactoryMethodFourthOrder();
00309     TestDefaultConstructorFactoryMethodSixthOrder();
00310     TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00311     ();
00312     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00313     ();
00314     TestReturnAsDenseMatrixWithGrid();
00315     TestReturnAsDimensionlessDenseMatrix();
00316     TestWriteToFile();
00317     TestMimBndy();
00318 }
```

18.149 tests/mtk_grad_2d_test/mtk_grad_2d_test.cc File Reference

Test file for the `mtk::Grad2D` class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_grad_2d_test.cc:
```



Functions

- void `TestDefaultConstructorFactory ()`
- void `TestReturnAsDenseMatrixWriteToFile ()`
- int `main ()`

18.149.1 Detailed Description

Author

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

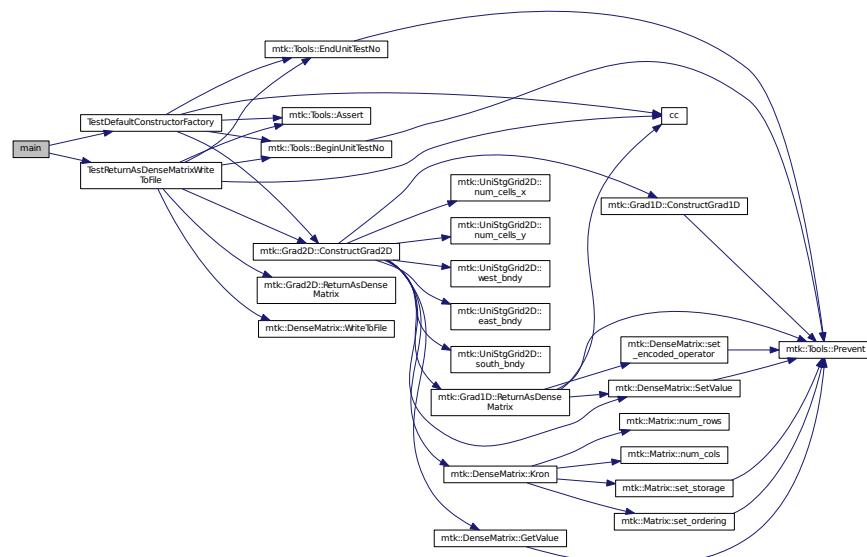
Definition in file [mtk_grad_2d_test.cc](#).

18.149.2 Function Documentation

18.149.2.1 int main ()

Definition at line 125 of file [mtk_grad_2d_test.cc](#).

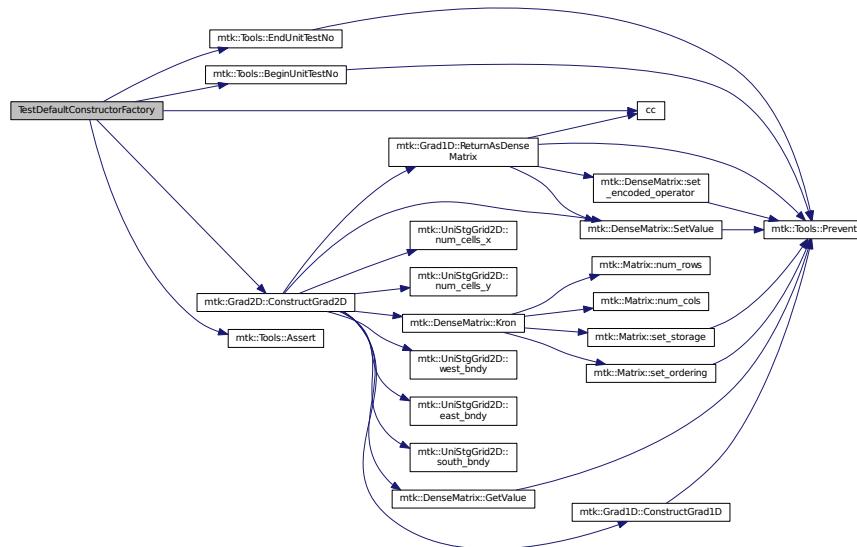
Here is the call graph for this function:



18.149.2.2 void TestDefaultConstructorFactory ()

Definition at line 61 of file [mtk_grad_2d_test.cc](#).

Here is the call graph for this function:



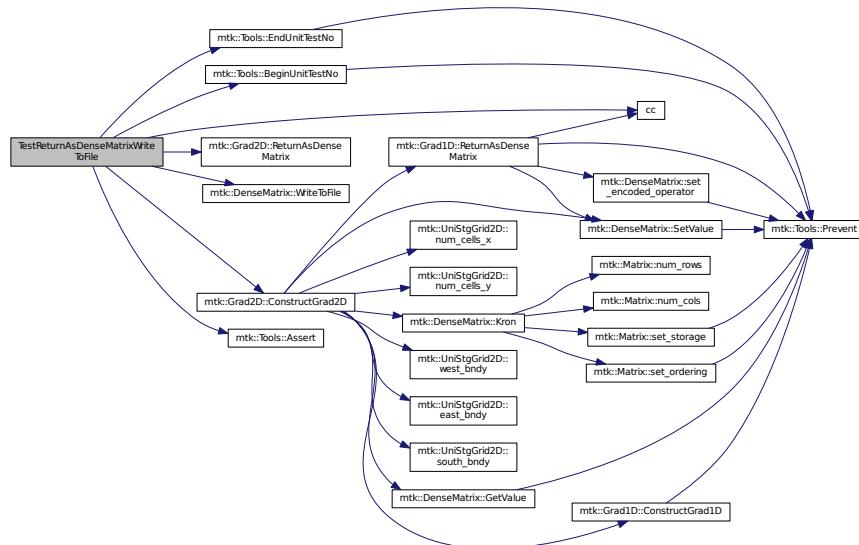
Here is the caller graph for this function:



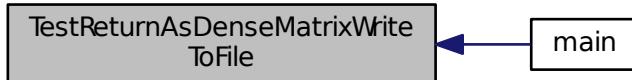
18.149.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 87 of file [mtk_grad_2d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.150 mtk_grad_2d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
  
```

```
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Grad2D gg;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real dd = 1.0;
00071
00072     int nn = 5;
00073     int mm = 5;
00074
00075     mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm,
00076                             mtk::FieldNature::VECTOR);
00077
00078     bool assertion = gg.ConstructGrad2D(ggg);
00079
00080     if (!assertion) {
00081         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00082     }
00083
00084     mtk::Tools::EndUnitTestNo(1);
00085     mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestReturnAsDenseMatrixWriteToFile() {
00089
00090     mtk::Tools::BeginUnitTestNo(2);
00091
00092     mtk::Grad2D gg;
00093
00094     mtk::Real aa = 0.0;
00095     mtk::Real bb = 1.0;
00096     mtk::Real cc = 0.0;
00097     mtk::Real dd = 1.0;
00098
00099     int nn = 5;
00100     int mm = 5;
00101
00102     mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm,
00103                             mtk::FieldNature::VECTOR);
00104
00105     bool assertion = gg.ConstructGrad2D(ggg);
00106
00107     if (!assertion) {
```

```

00106     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00107 }
00108
00109 mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00110
00111 assertion = assertion && (ggm.num_rows() != mtk::kZero);
00112
00113 std::cout << ggm << std::endl;
00114
00115 assertion = assertion && ggm.WriteToFile("mtk_grad_2d_test_02_ggm.dat");
00116
00117 if(!assertion) {
00118     std::cerr << "Error writing to file." << std::endl;
00119 }
00120
00121 mtk::Tools::EndUnitTestNo(2);
00122 mtk::Tools::Assert(assertion);
00123 }
00124
00125 int main () {
00126
00127     std::cout << "Testing mtk::Grad2D class." << std::endl;
00128
00129     TestDefaultConstructorFactory();
00130     TestReturnAsDenseMatrixWriteToFile();
00131 }
```

18.151 tests/mtk_grad_3d_test/mtk_grad_3d_test.cc File Reference

Test file for the `mtk::Grad3D` class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_grad_3d_test.cc:
```



Functions

- void `TestDefaultConstructorFactory ()`
- void `TestReturnAsDenseMatrixWriteToFile ()`
- int `main ()`

18.151.1 Detailed Description

Author

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

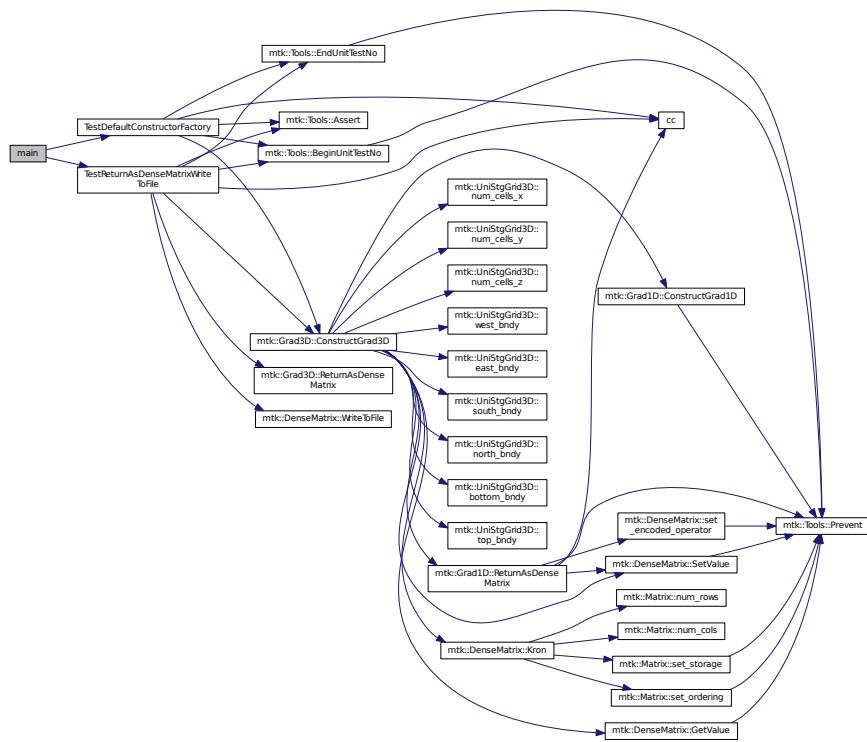
Definition in file [mtk_grad_3d_test.cc](#).

18.151.2 Function Documentation

18.151.2.1 int main ()

Definition at line 133 of file [mtk_grad_3d_test.cc](#).

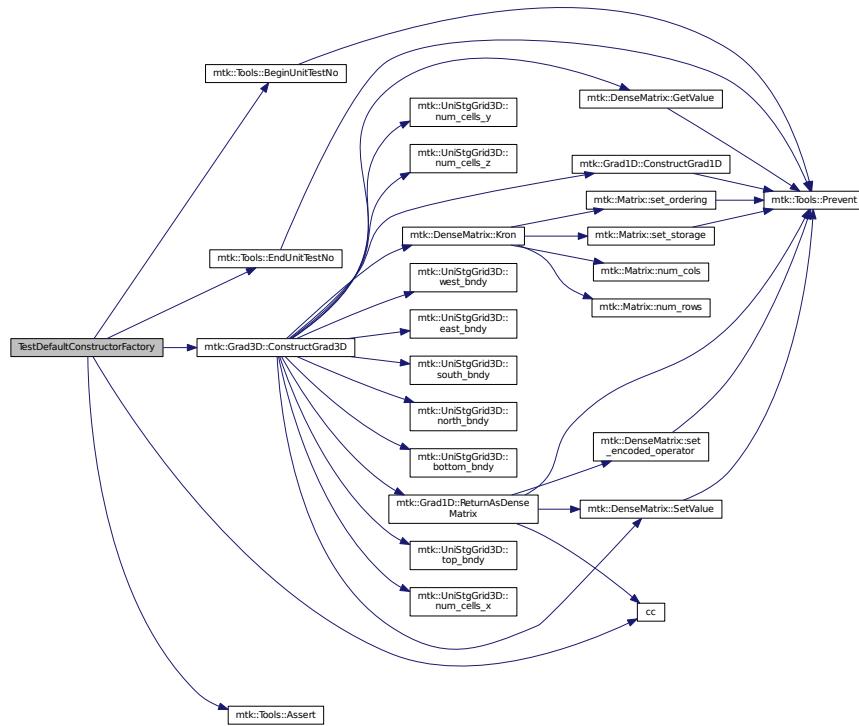
Here is the call graph for this function:



18.151.2.2 void TestDefaultConstructorFactory ()

Definition at line 61 of file [mtk_grad_3d_test.cc](#).

Here is the call graph for this function:



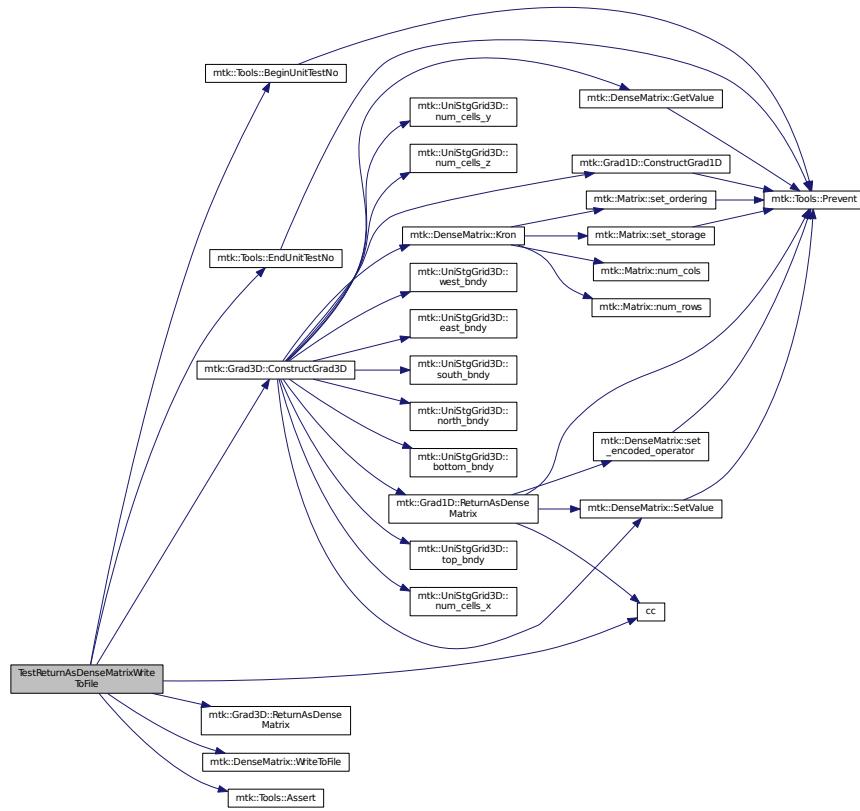
Here is the caller graph for this function:



18.151.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 91 of file [mtk_grad_3d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.152 mtk_grad_3d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
  
```

```
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csdc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csdc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Grad3D gg;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real dd = 1.0;
00071     mtk::Real ee = 0.0;
00072     mtk::Real ff = 1.0;
00073
00074     int nn = 5;
00075     int mm = 5;
00076     int oo = 5;
00077
00078     mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo,
00079     mtk::FieldNature::VECTOR);
00080
00081     bool assertion = gg.ConstructGrad3D(ggg);
00082
00083     if (!assertion) {
00084         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00085     }
00086
00087     mtk::Tools::EndUnitTestNo(1);
00088     mtk::Tools::Assert(assertion);
00089 }
00090
00091 void TestReturnAsDenseMatrixWriteToFile() {
00092
00093     mtk::Tools::BeginUnitTestNo(2);
00094
00095     mtk::Grad3D gg;
```

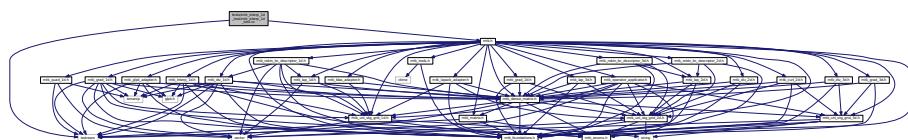
```

00096
00097     mtk::Real aa = 0.0;
00098     mtk::Real bb = 1.0;
00099     mtk::Real cc = 0.0;
00100    mtk::Real dd = 1.0;
00101    mtk::Real ee = 0.0;
00102    mtk::Real ff = 1.0;
00103
00104    int nn = 5;
00105    int mm = 5;
00106    int oo = 5;
00107
00108    mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo,
00109    mtk::FieldNature::VECTOR);
00110
00111    bool assertion = gg.ConstructGrad3D(ggg);
00112
00113    if (!assertion) {
00114        std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00115    }
00116
00117    mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00118
00119    assertion = assertion && (ggm.num_rows() != mtk::kZero);
00120
00121    std::cout << ggm << std::endl;
00122
00123    assertion = assertion && ggm.WriteToFile("mtk_grad_3d_test_02_ggm.dat");
00124
00125    if(!assertion) {
00126        std::cerr << "Error writing to file." << std::endl;
00127    }
00128
00129    mtk::Tools::EndUnitTestNo(2);
00130    mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135    std::cout << "Testing mtk::Grad2D class." << std::endl;
00136
00137    TestDefaultConstructorFactory();
00138    TestReturnAsDenseMatrixWriteToFile();
00139 }
```

18.153 tests/mtk_interp_1d_test/mtk_interp_1d_test.cc File Reference

Testing the 1D interpolation.

```
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_interp_1d_test.cc:
```



Functions

- void `TestDefaultConstructorFactoryMethodDefault ()`
- void `TestReturnAsDenseMatrixWithGrid ()`
- int `main ()`

18.153.1 Detailed Description

Author

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

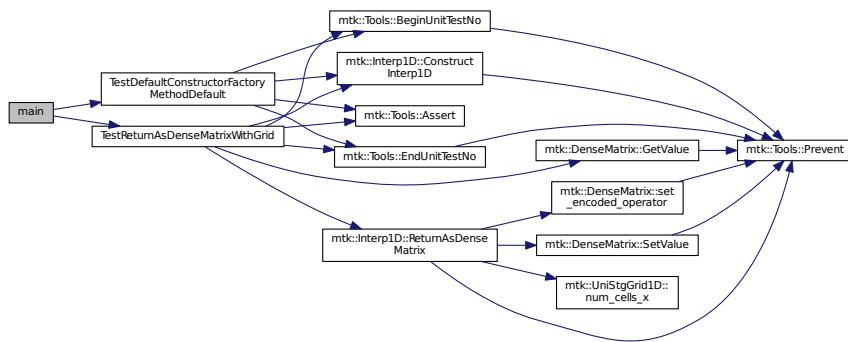
Definition in file [mtk_interp_1d_test.cc](#).

18.153.2 Function Documentation

18.153.2.1 int main ()

Definition at line 99 of file [mtk_interp_1d_test.cc](#).

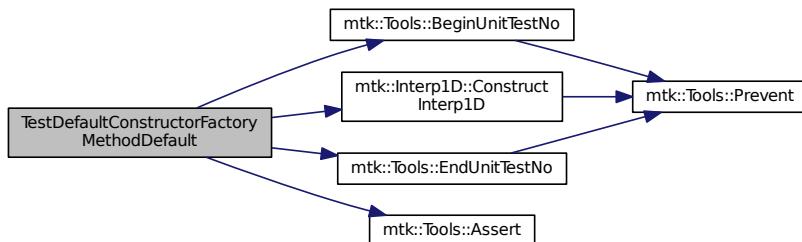
Here is the call graph for this function:



18.153.2.2 void TestDefaultConstructorFactoryMethodDefault ()

Definition at line 60 of file [mtk_interp_1d_test.cc](#).

Here is the call graph for this function:



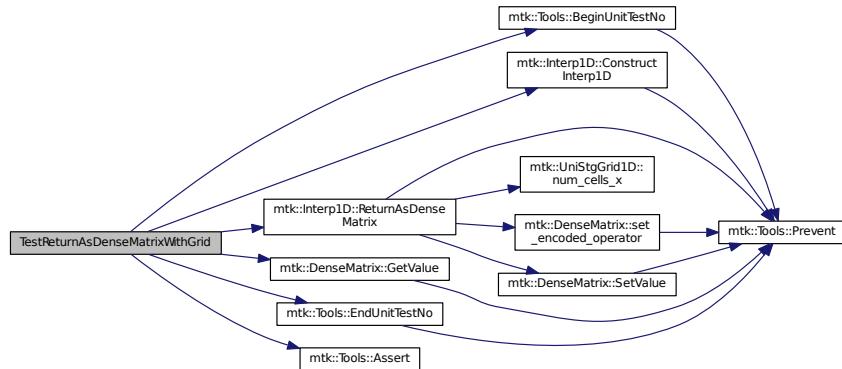
Here is the caller graph for this function:



18.153.2.3 void TestReturnAsDenseMatrixWithGrid ()

Definition at line 76 of file [mtk_interp_1d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.154 mtk_interp_1d_test.cc

00001

```

00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062     mtk::Tools::BeginUnitTestNo(1);
00063
00064     mtk::Interp1D inter;
00065
00066     bool assertion = inter.ConstructInterp1D();
00067
00068     if (!assertion) {
00069         std::cerr << "Mimetic interp could not be built." << std::endl;
00070     }
00071
00072     mtk::Tools::EndUnitTestNo(1);
00073     mtk::Tools::Assert(assertion);
00074 }
00075
00076 void TestReturnAsDenseMatrixWithGrid() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Interp1D inter;
00081
00082     bool assertion = inter.ConstructInterp1D();
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00086     }
00087
00088     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00089
00090     mtk::DenseMatrix interpm(inter.ReturnAsDenseMatrix(grid));

```

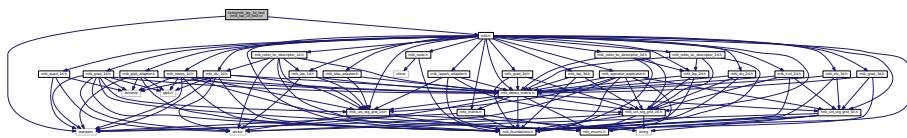
```

00091     assertion =
00092     assertion && interpm.GetValue(0,0) == 1.0 && interpm.GetValue(5,6) == 1.0;
00094
00095     mtk::Tools::EndUnitTestNo(2);
00096     mtk::Tools::Assert(assertion);
00097 }
00098
00099 int main () {
00100     std::cout << "Testing mtk::Interp1D class." << std::endl;
00102
00103     TestDefaultConstructorFactoryMethodDefault();
00104     TestReturnAsDenseMatrixWithGrid();
00105 }
```

18.155 tests/mtk_lap_1d_test/mtk_lap_1d_test.cc File Reference

Testing the 1D Laplacian operator.

```
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_lap_1d_test.cc:
```



Functions

- void [TestDefaultConstructorFactoryMethodDefault\(\)](#)
- void [TestDefaultConstructorFactoryMethodFourthOrder\(\)](#)
- void [TestDefaultConstructorFactoryMethodSixthOrder\(\)](#)
- void [TestDefaultConstructorFactoryMethodEightOrderDefThreshold\(\)](#)
- void [TestDefaultConstructorFactoryMethodTenthOrderDefThreshold\(\)](#)
- void [TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold\(\)](#)
- void [TestReturnAsDenseMatrix\(\)](#)
- int [main\(\)](#)

18.155.1 Detailed Description

Author

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

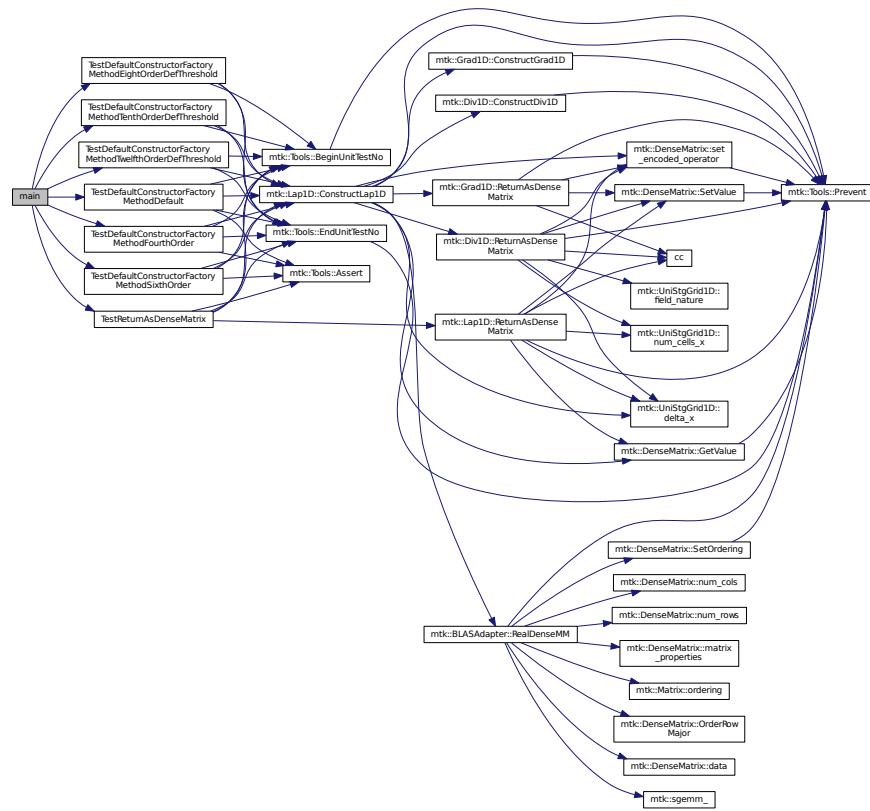
Definition in file [mtk_lap_1d_test.cc](#).

18.155.2 Function Documentation

18.155.2.1 int main ()

Definition at line 176 of file [mtk_lap_1d_test.cc](#).

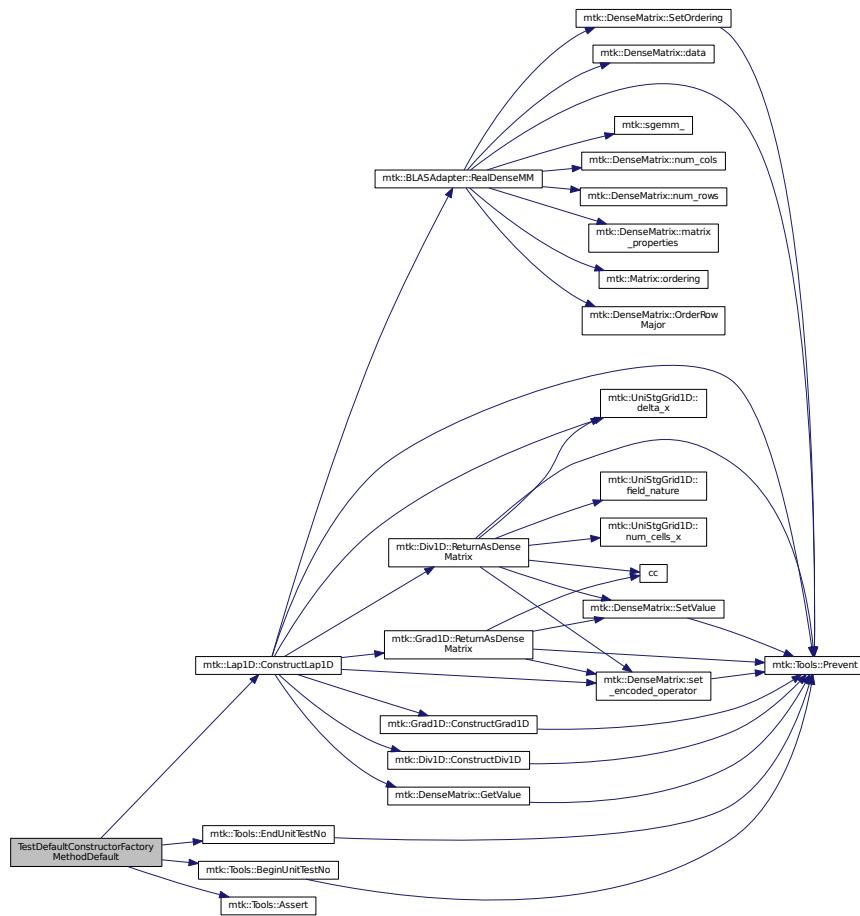
Here is the call graph for this function:



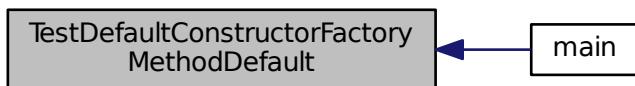
18.155.2.2 void TestDefaultConstructorFactoryMethodDefault()

Definition at line 60 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



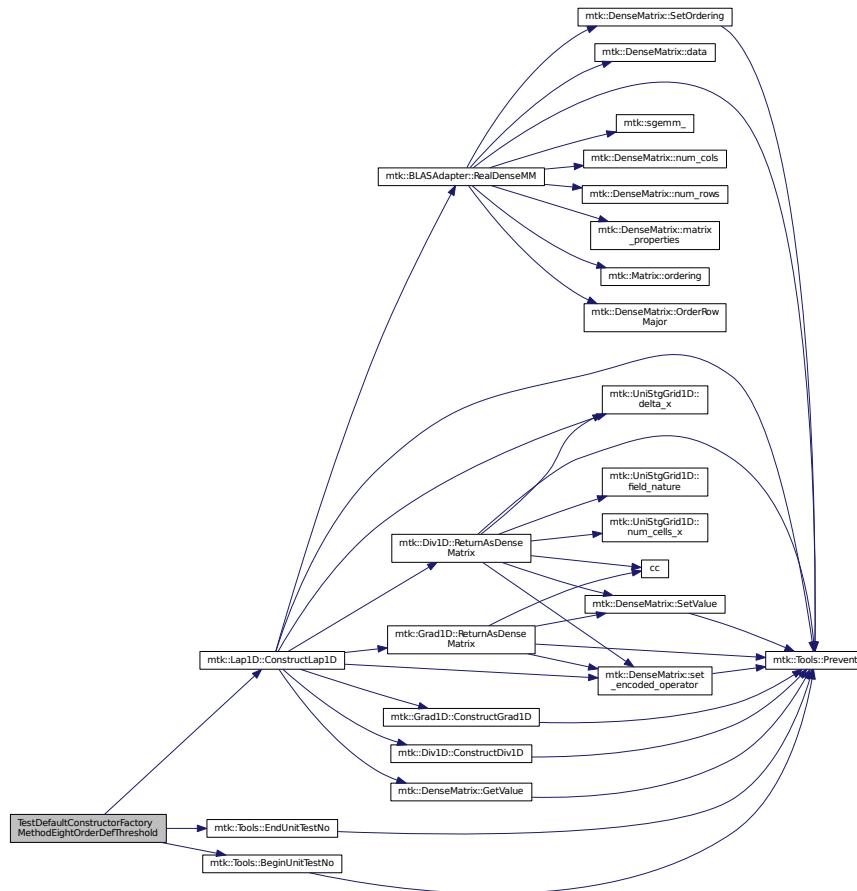
Here is the caller graph for this function:



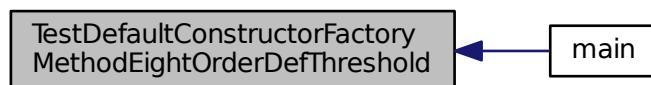
18.155.2.3 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold()

Definition at line 108 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



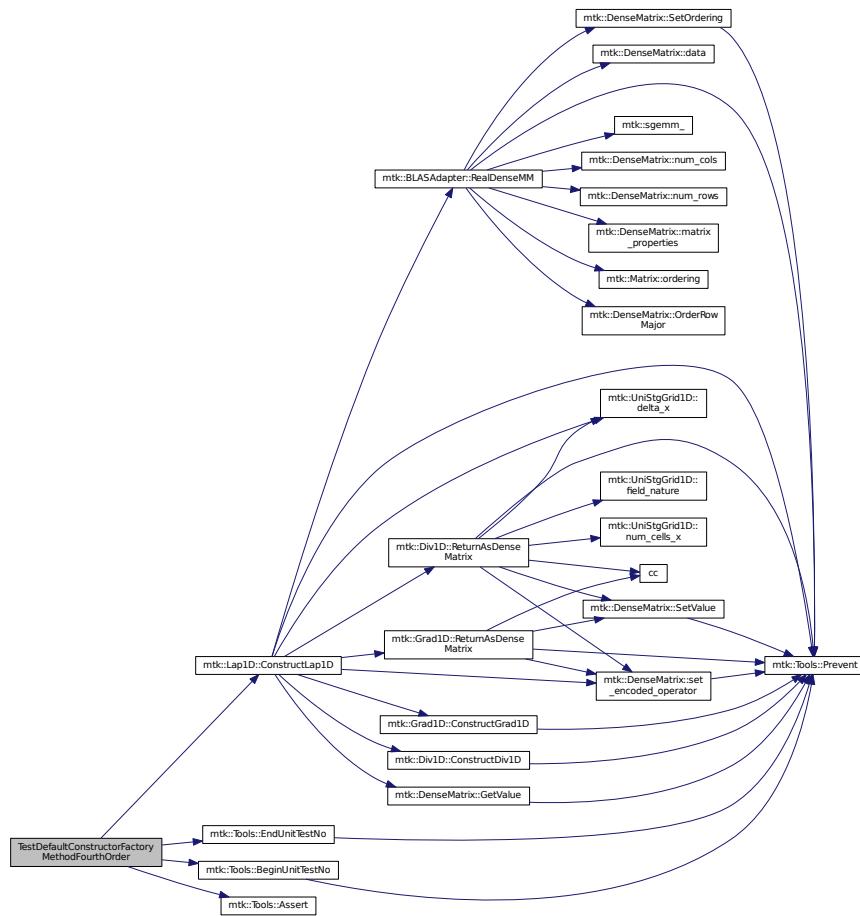
Here is the caller graph for this function:



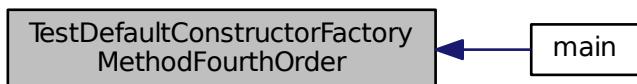
18.155.2.4 void TestDefaultConstructorFactoryMethodFourthOrder()

Definition at line 76 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



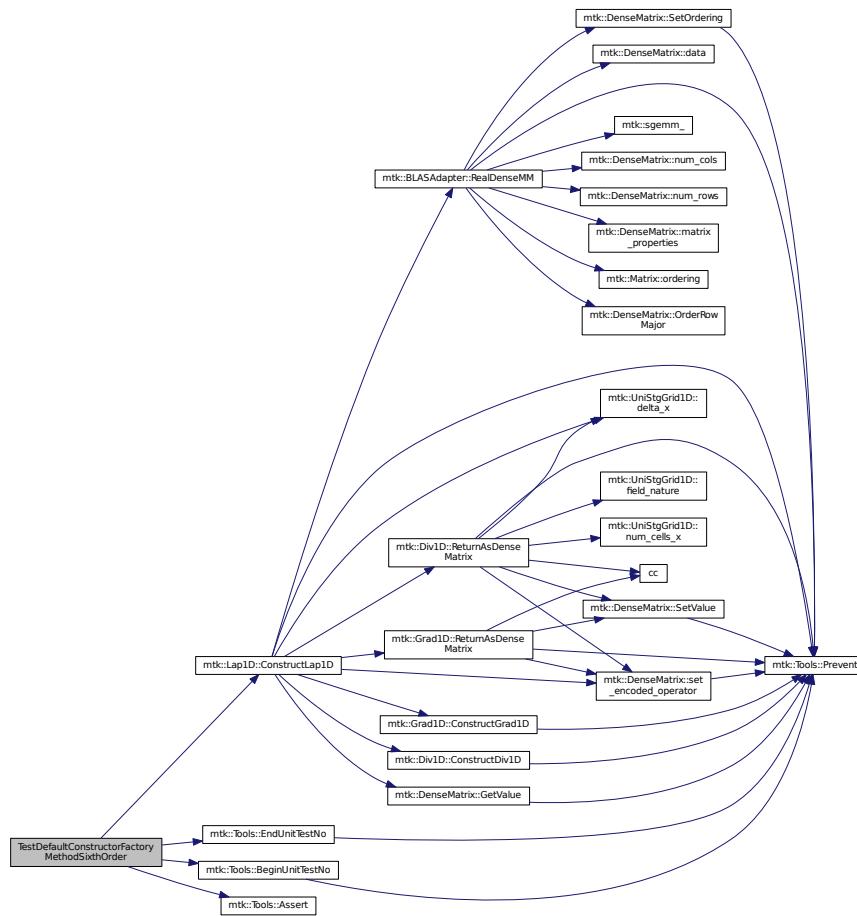
Here is the caller graph for this function:



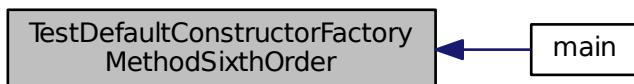
18.155.2.5 void TestDefaultConstructorFactoryMethodSixthOrder()

Definition at line 92 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



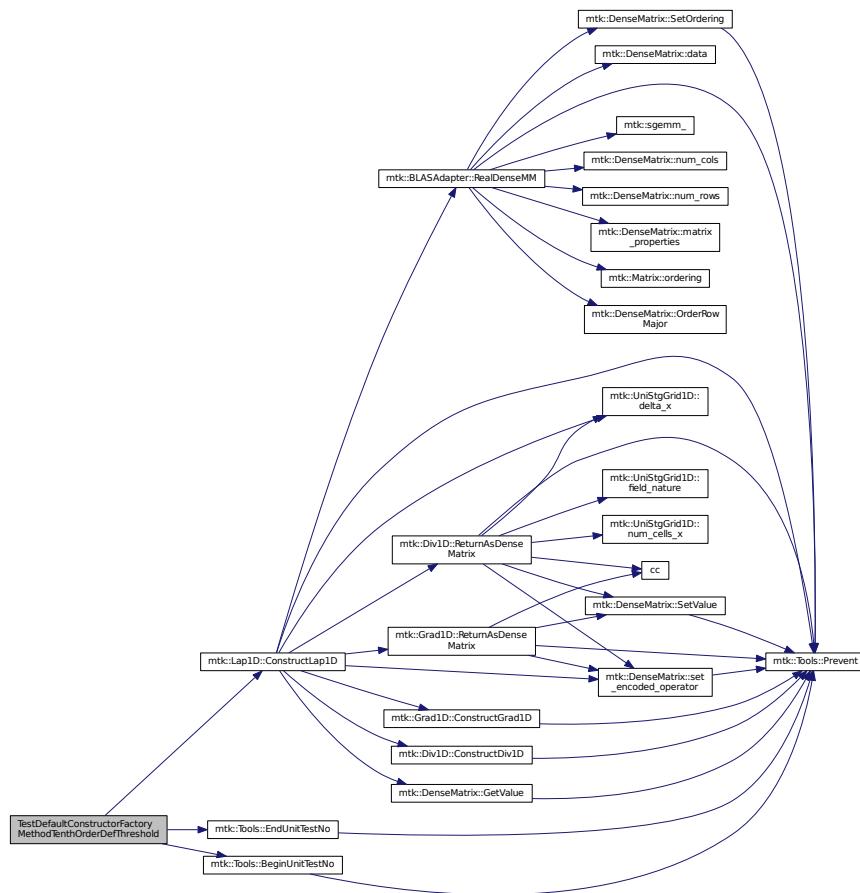
Here is the caller graph for this function:



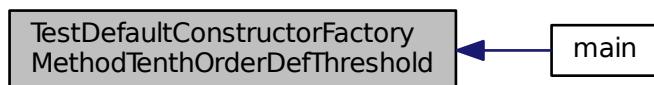
18.155.2.6 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold()

Definition at line 123 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



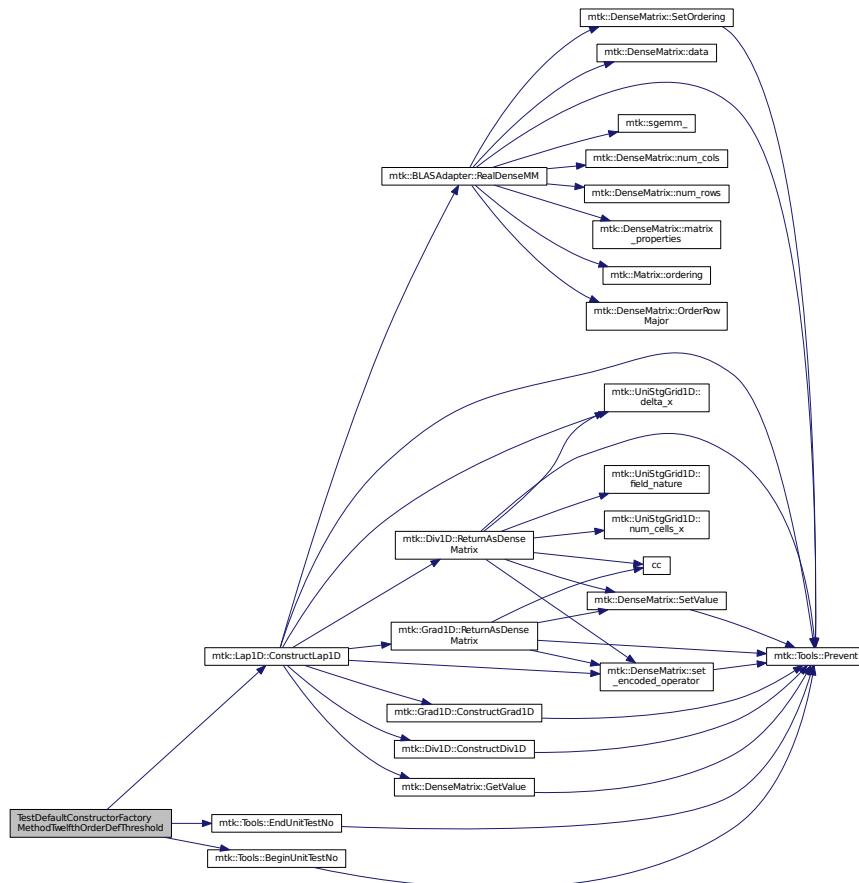
Here is the caller graph for this function:



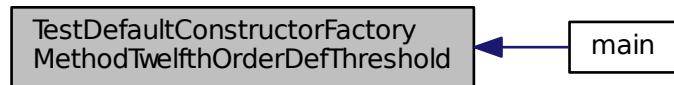
18.155.2.7 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold()

Definition at line 138 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



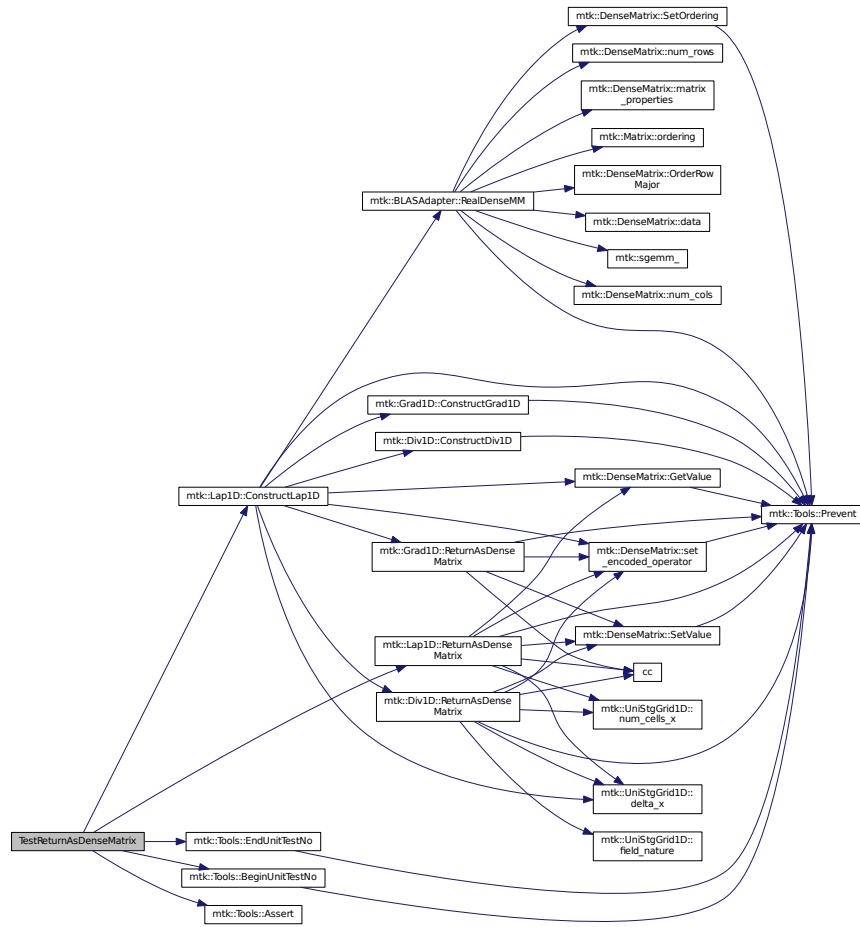
Here is the caller graph for this function:



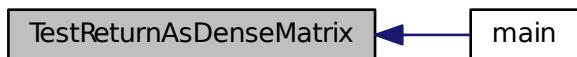
18.155.2.8 void TestReturnAsDenseMatrix()

Definition at line 153 of file [mtk_lap_1d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.156 mtk_lap_1d_test.cc

```

00001
00010 /*
00011 Copyright (C) 2016, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
  
```

```
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062     mtk::Tools::BeginUnitTestNo(1);
00063
00064     mtk::Lap1D lap2;
00065
00066     bool assertion = lap2.ConstructLap1D();
00067
00068     if (!assertion) {
00069         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00070     }
00071
00072     mtk::Tools::EndUnitTestNo(1);
00073     mtk::Tools::Assert(assertion);
00074 }
00075
00076 void TestDefaultConstructorFactoryMethodFourthOrder() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Lap1D lap4;
00081
00082     bool assertion = lap4.ConstructLap1D(4);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00086     }
00087
00088     mtk::Tools::EndUnitTestNo(2);
00089     mtk::Tools::Assert(assertion);
00090 }
00091
00092 void TestDefaultConstructorFactoryMethodSixthOrder() {
00093 }
```

```
00094     mtk::Tools::BeginUnitTestNo(3);
00095
00096     mtk::Lap1D lap6;
00097
00098     bool assertion = lap6.ConstructLap1D(6);
00099
00100    if (!assertion) {
00101        std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00102    }
00103
00104    mtk::Tools::EndUnitTestNo(3);
00105    mtk::Tools::Assert(assertion);
00106 }
00107
00108 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00109 () {
00110     mtk::Tools::BeginUnitTestNo(4);
00111
00112     mtk::Lap1D lap8;
00113
00114     bool assertion = lap8.ConstructLap1D(8);
00115
00116    if (!assertion) {
00117        std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00118    }
00119
00120    mtk::Tools::EndUnitTestNo(4);
00121 }
00122
00123 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00124 () {
00125     mtk::Tools::BeginUnitTestNo(5);
00126
00127     mtk::Lap1D lap10;
00128
00129     bool assertion = lap10.ConstructLap1D(10);
00130
00131    if (!assertion) {
00132        std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00133    }
00134
00135    mtk::Tools::EndUnitTestNo(5);
00136 }
00137
00138 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold
00139 () {
00140     mtk::Tools::BeginUnitTestNo(6);
00141
00142     mtk::Lap1D lap12;
00143
00144     bool assertion = lap12.ConstructLap1D(12);
00145
00146    if (!assertion) {
00147        std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00148    }
00149
00150    mtk::Tools::EndUnitTestNo(6);
00151 }
00152
00153 void TestReturnAsDenseMatrix() {
00154
00155     mtk::Tools::BeginUnitTestNo(8);
00156
00157     mtk::Lap1D lap4;
00158
00159     bool assertion = lap4.ConstructLap1D(4);
00160
00161    if (!assertion) {
00162        std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00163    }
00164
00165     mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00166
00167     mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00168
00169     assertion = assertion &&
00170         abs(lap4_m.GetValue(1, 0) - 385.133) < mtk::kDefaultTolerance &&
00171         abs(lap4_m.GetValue(11, 12) - 385.133) < mtk::kDefaultTolerance;
```

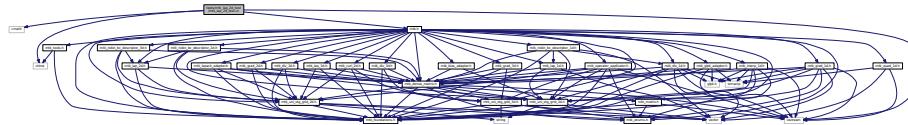
```

00172     mtk::Tools::EndUnitTestNo(8);
00173     mtk::Tools::Assert(assertion);
00174 }
00175
00176 int main () {
00177     std::cout << "Testing MTK 1D Laplacian" << std::endl;
00178     TestDefaultConstructorFactoryMethodDefault();
00180     TestDefaultConstructorFactoryMethodFourthOrder();
00181     TestDefaultConstructorFactoryMethodSixthOrder();
00182     TestDefaultConstructorFactoryMethodEightOrderDefThreshold
00183     ();
00184     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold
00185     ();
00186     TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold
00187     ();
00186     TestReturnAsDenseMatrix();
00187 }
```

18.157 tests/mtk_lap_2d_test/mtk_lap_2d_test.cc File Reference

Test file for the `mtk::Lap2D` class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_lap_2d_test.cc:
```



Functions

- void `TestDefaultConstructorFactory ()`
- void `TestReturnAsDenseMatrixWriteToFile ()`
- int `main ()`

18.157.1 Detailed Description

Author

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

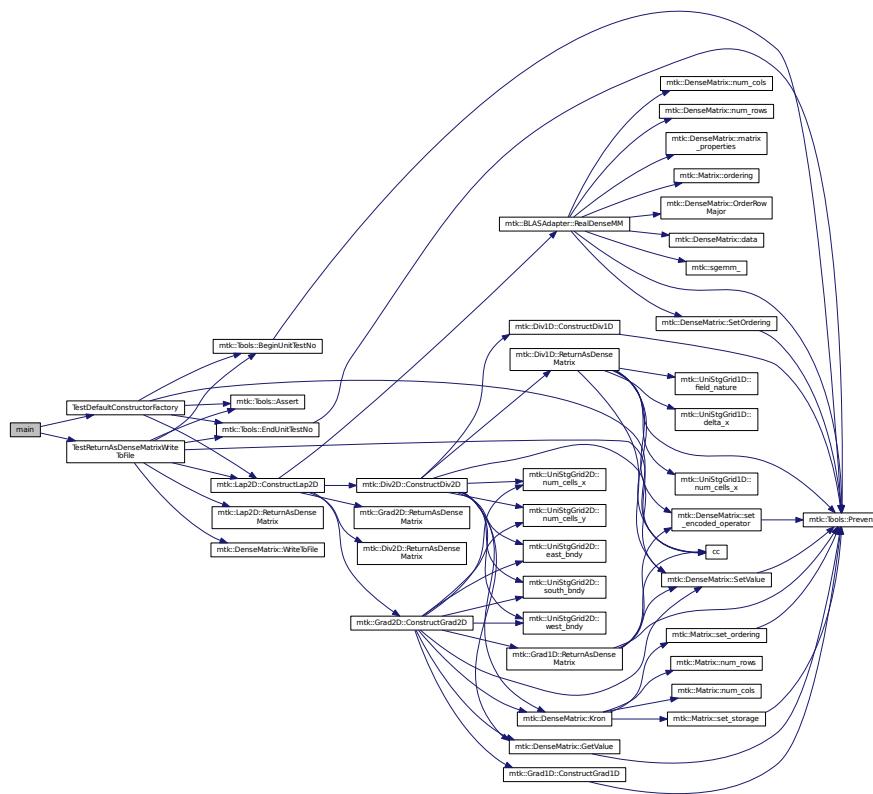
Definition in file [mtk_lap_2d_test.cc](#).

18.157.2 Function Documentation

18.157.2.1 int main ()

Definition at line 125 of file [mtk_lap_2d_test.cc](#).

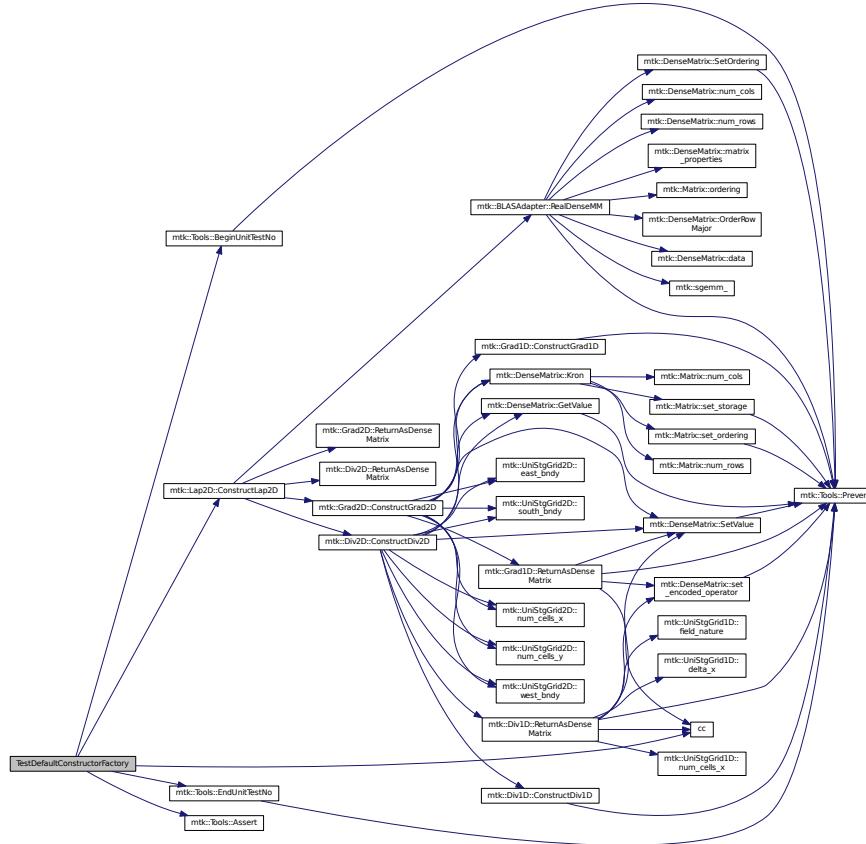
Here is the call graph for this function:



18.157.2.2 void TestDefaultConstructorFactory()

Definition at line 61 of file [mtk_lap_2d_test.cc](#).

Here is the call graph for this function:



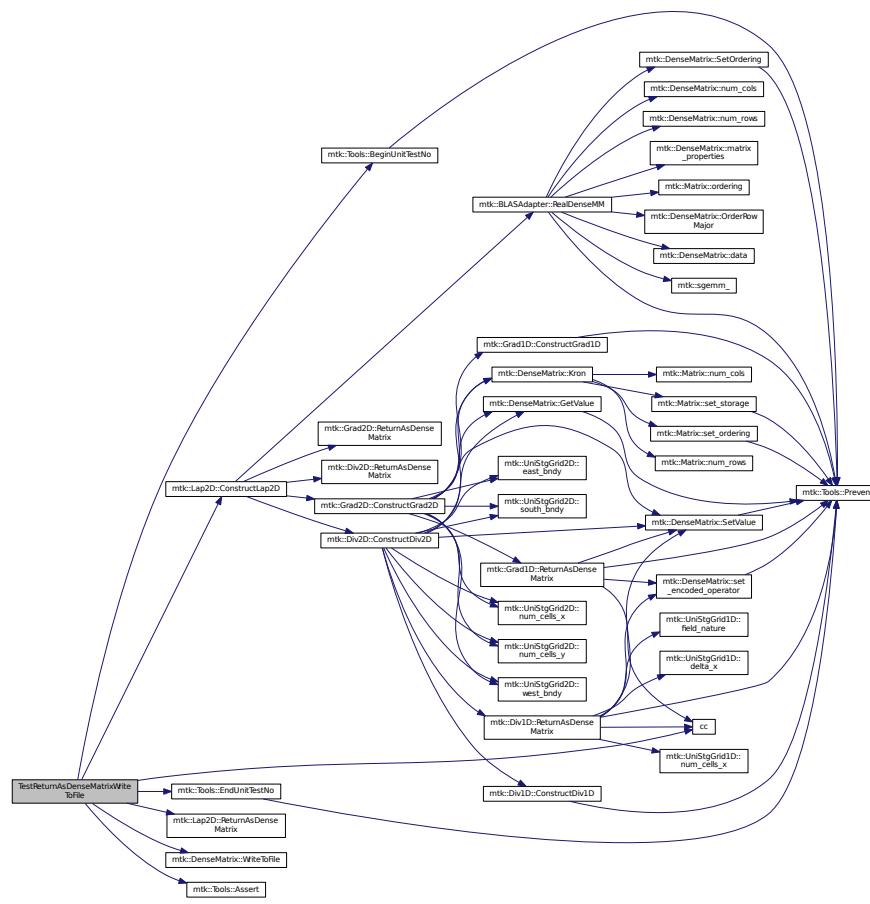
Here is the caller graph for this function:



18.157.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 87 of file [mtk_lap_2d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.158 mtk_lap_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
```

```

00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Lap2D ll;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real dd = 1.0;
00071
00072     int nn = 5;
00073     int mm = 5;
00074
00075     mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00076
00077     bool assertion = ll.ConstructLap2D(llg);
00078
00079     if (!assertion) {
00080         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00081     }
00082
00083     mtk::Tools::EndUnitTestNo(1);
00084     mtk::Tools::Assert(assertion);
00085 }
00086
00087 void TestReturnAsDenseMatrixWriteToFile() {
00088
00089     mtk::Tools::BeginUnitTestNo(2);
00090
00091     mtk::Lap2D ll;
00092

```

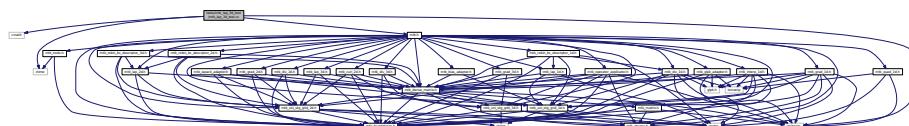
```

00093     mtk::Real aa = 0.0;
00094     mtk::Real bb = 1.0;
00095     mtk::Real cc = 0.0;
00096     mtk::Real dd = 1.0;
00097
00098     int nn = 5;
00099     int mm = 5;
00100
00101     mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00102
00103     bool assertion = llg.ConstructLap2D();
00104
00105     if (!assertion) {
00106         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00107     }
00108
00109     mtk::DenseMatrix l1m(llg.ReturnAsDenseMatrix());
00110
00111     assertion = assertion && (l1m.num_rows() != 0);
00112
00113     std::cout << l1m << std::endl;
00114
00115     assertion = assertion && l1m.WriteToFile("mtk_lap_2d_test_02.dat");
00116
00117     if (!assertion) {
00118         std::cerr << "Error writing to file." << std::endl;
00119     }
00120
00121     mtk::Tools::EndUnitTestNo(2);
00122     mtk::Tools::Assert(assertion);
00123 }
00124
00125 int main () {
00126
00127     std::cout << "Testing mtk::Lap2D class." << std::endl;
00128
00129     TestDefaultConstructorFactory();
00130     TestReturnAsDenseMatrixWriteToFile();
00131 }
```

18.159 tests/mtk_lap_3d_test/mtk_lap_3d_test.cc File Reference

Test file for the `mtk::Lap3D` class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_lap_3d_test.cc:
```



Functions

- void `TestDefaultConstructorFactory ()`
- void `TestReturnAsDenseMatrixWriteToFile ()`
- int `main ()`

18.159.1 Detailed Description

Author

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

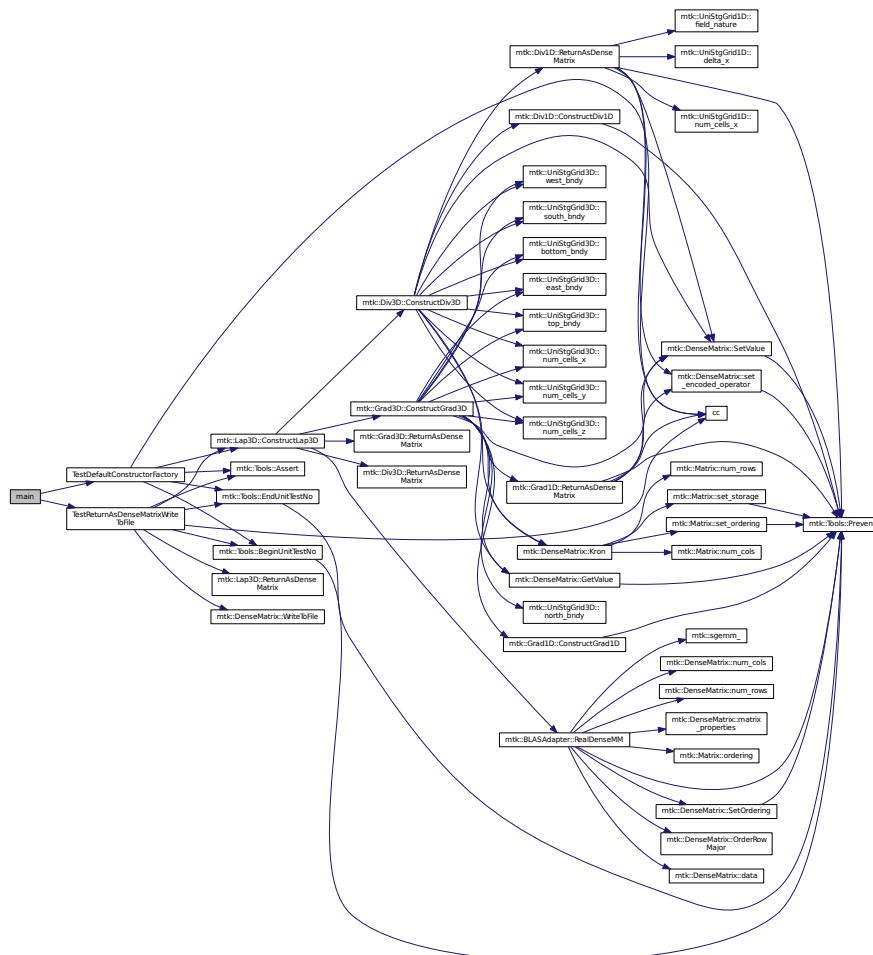
Definition in file [mtk_lap_3d_test.cc](#).

18.159.2 Function Documentation

18.159.2.1 int main()

Definition at line 131 of file [mtk_lap_3d_test.cc](#).

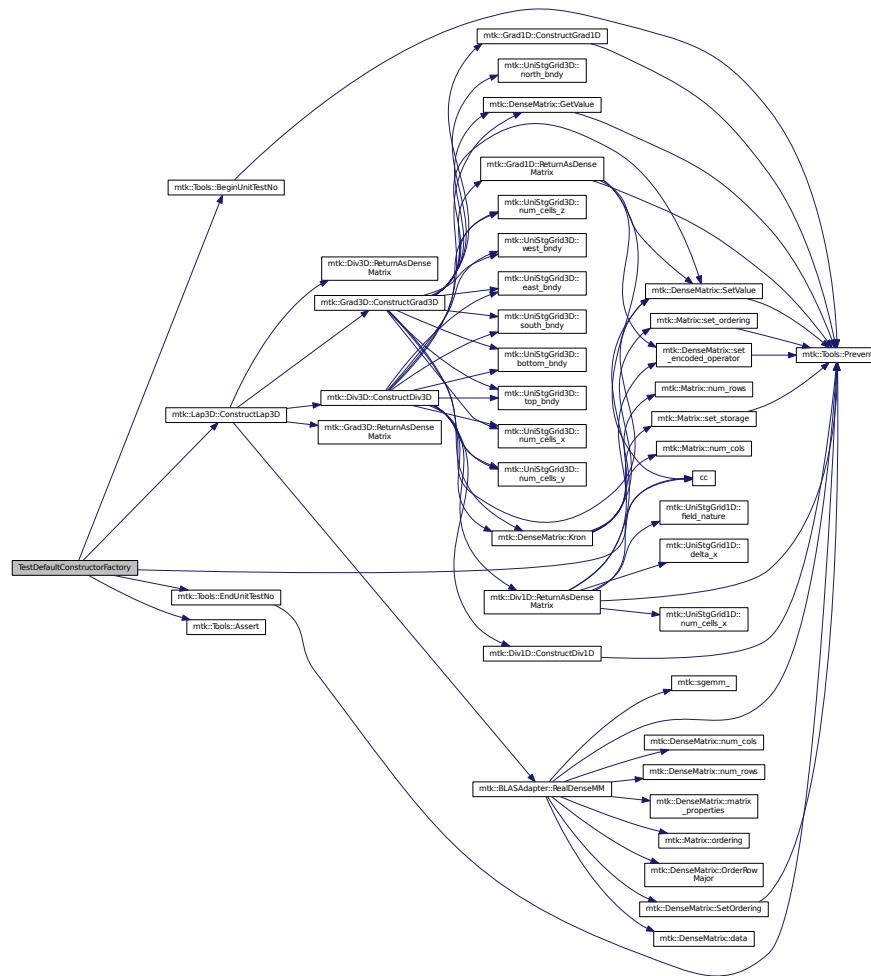
Here is the call graph for this function:



18.159.2.2 void TestDefaultConstructorFactory()

Definition at line 61 of file [mtk_lap_3d_test.cc](#).

Here is the call graph for this function:



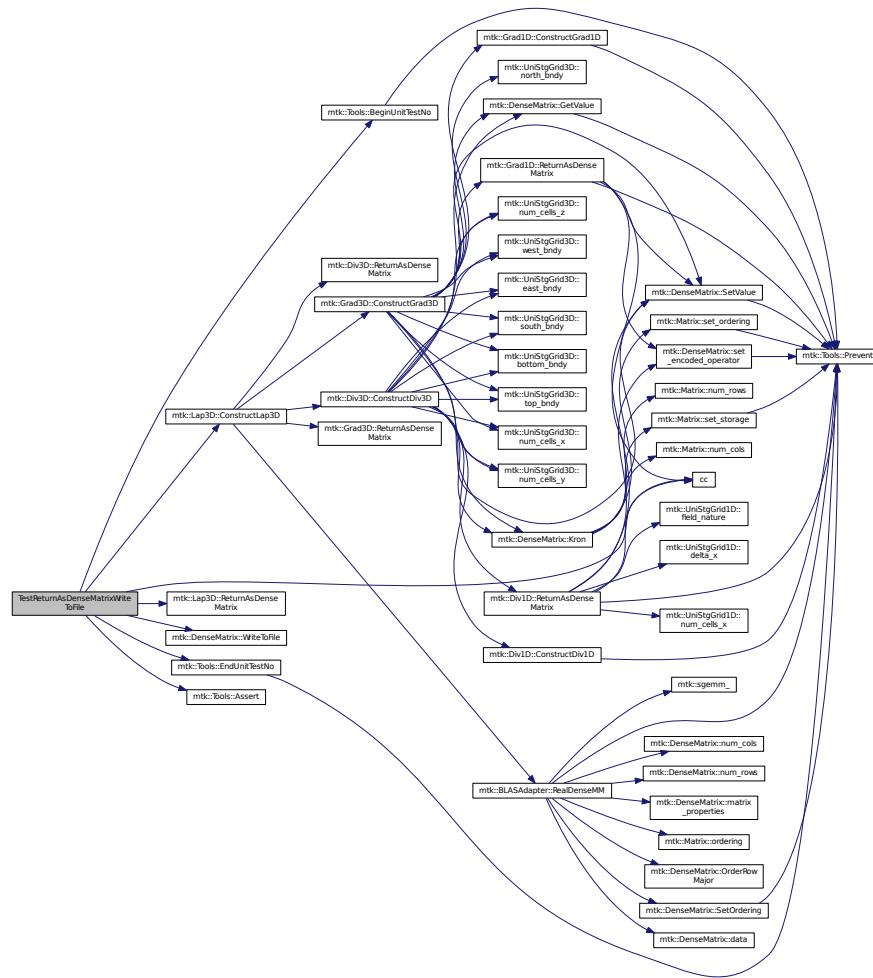
Here is the caller graph for this function:



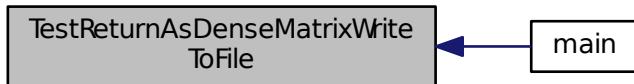
18.159.2.3 void TestReturnAsDenseMatrixWriteToFile()

Definition at line 90 of file [mtk_lap_3d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.160 mtk_lap_3d_test.cc

00001

```
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorFactory() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Lap3D ll;
00066
00067     mtk::Real aa = 0.0;
00068     mtk::Real bb = 1.0;
00069     mtk::Real cc = 0.0;
00070     mtk::Real dd = 1.0;
00071     mtk::Real ee = 0.0;
00072     mtk::Real ff = 1.0;
00073
00074     int nn = 5;
00075     int mm = 5;
00076     int oo = 5;
00077
00078     mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00079
00080     bool assertion = ll.ConstructLap3D(llg);
00081
00082     if (!assertion) {
00083         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00084     }
00085
00086     mtk::Tools::EndUnitTestNo(1);
00087     mtk::Tools::Assert(assertion);
00088 }
```

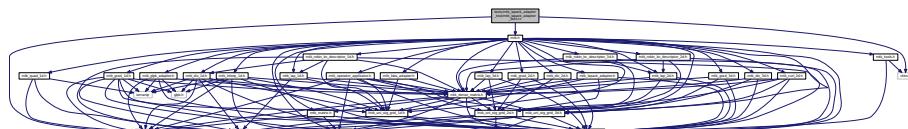
```

00089
00090 void TestReturnAsDenseMatrixWriteToFile() {
00091
00092     mtk::Tools::BeginUnitTestNo(2);
00093
00094     mtk::Lap3D ll;
00095
00096     mtk::Real aa = 0.0;
00097     mtk::Real bb = 1.0;
00098     mtk::Real cc = 0.0;
00099     mtk::Real dd = 1.0;
00100    mtk::Real ee = 0.0;
00101    mtk::Real ff = 1.0;
00102
00103    int nn = 5;
00104    int mm = 5;
00105    int oo = 5;
00106
00107    mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00108
00109    bool assertion = ll.ConstructLap3D(llg);
00110
00111    if (!assertion) {
00112        std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00113    }
00114
00115    mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00116
00117    assertion = assertion && (llm.num_rows() != 0);
00118
00119    std::cout << llm << std::endl;
00120
00121    assertion = assertion && llm.WriteToFile("mtk_lap_3d_test_02_llm.dat");
00122
00123    if (!assertion) {
00124        std::cerr << "Error writing to file." << std::endl;
00125    }
00126
00127    mtk::Tools::EndUnitTestNo(2);
00128    mtk::Tools::Assert(assertion);
00129 }
00130
00131 int main () {
00132
00133    std::cout << "Testing mtk::Lap3D class." << std::endl;
00134
00135    TestDefaultConstructorFactory();
00136    TestReturnAsDenseMatrixWriteToFile();
00137 }
```

18.161 tests/mtk_lapack_adapter_test/mtk_lapack_adapter_test.cc File Reference

Test file for the `mtk::LAPACKAdapter` class.

```
#include <iostream>
#include <ctime>
#include "mtk.h"
Include dependency graph for mtk_lapack_adapter_test.cc:
```



Functions

- void `Test1()`

- int [main \(\)](#)

18.161.1 Detailed Description

Author

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

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

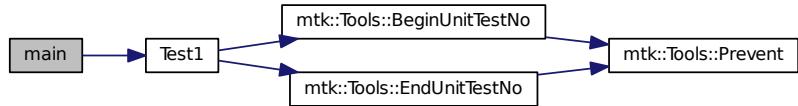
Definition in file [mtk_lapack_adapter_test.cc](#).

18.161.2 Function Documentation

18.161.2.1 int main ()

Definition at line [68](#) of file [mtk_lapack_adapter_test.cc](#).

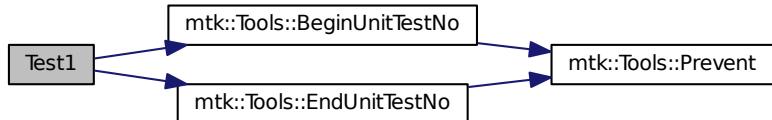
Here is the call graph for this function:



18.161.2.2 void Test1 ()

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

Here is the call graph for this function:



Here is the caller graph for this function:



18.162 mtk_lapack_adapter_test.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void Test1() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::Tools::EndUnitTestNo(1);

```

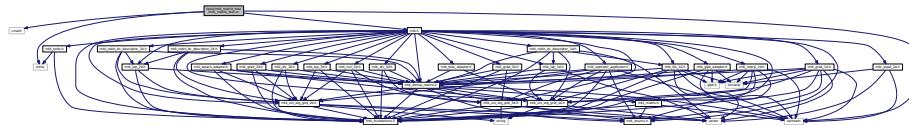
```

00066 }
00067
00068 int main () {
00069
00070     std::cout << "Testing mtk::LAPACKAdapter class." << std::endl;
00071
00072     Test1 ();
00073 }
```

18.163 tests/mtk_matrix_test/mtk_matrix_test.cc File Reference

Unit test file for the [mtk::Matrix](#) class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_matrix_test.cc:
```



Functions

- void [TestDefaultConstructorGetters \(\)](#)
- void [TestSettersGettersCopyConstructor \(\)](#)
- void [TestIncreaseDecreaseNumNull \(\)](#)
- void [TestIncreaseDecreaseNumZero \(\)](#)
- int [main \(int argc, char *argv\[\]\)](#)

18.163.1 Detailed Description

Author

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

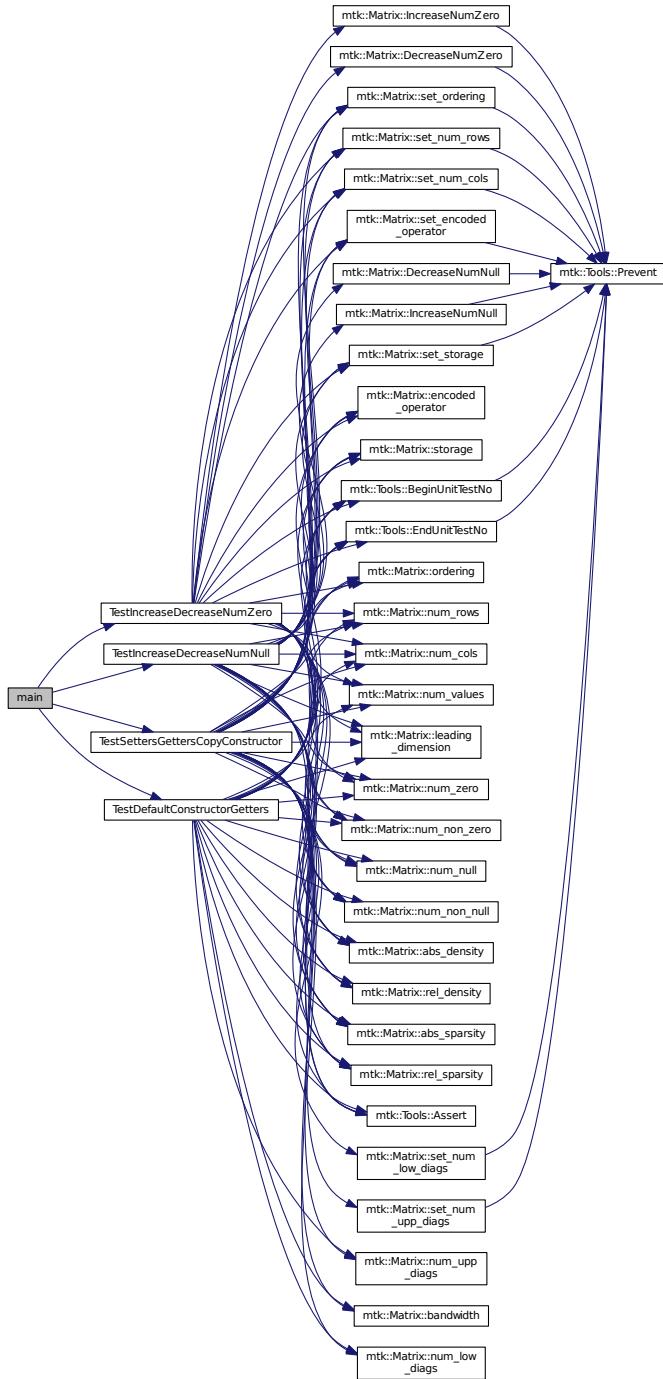
Definition in file [mtk_matrix_test.cc](#).

18.163.2 Function Documentation

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

Definition at line 300 of file [mtk_matrix_test.cc](#).

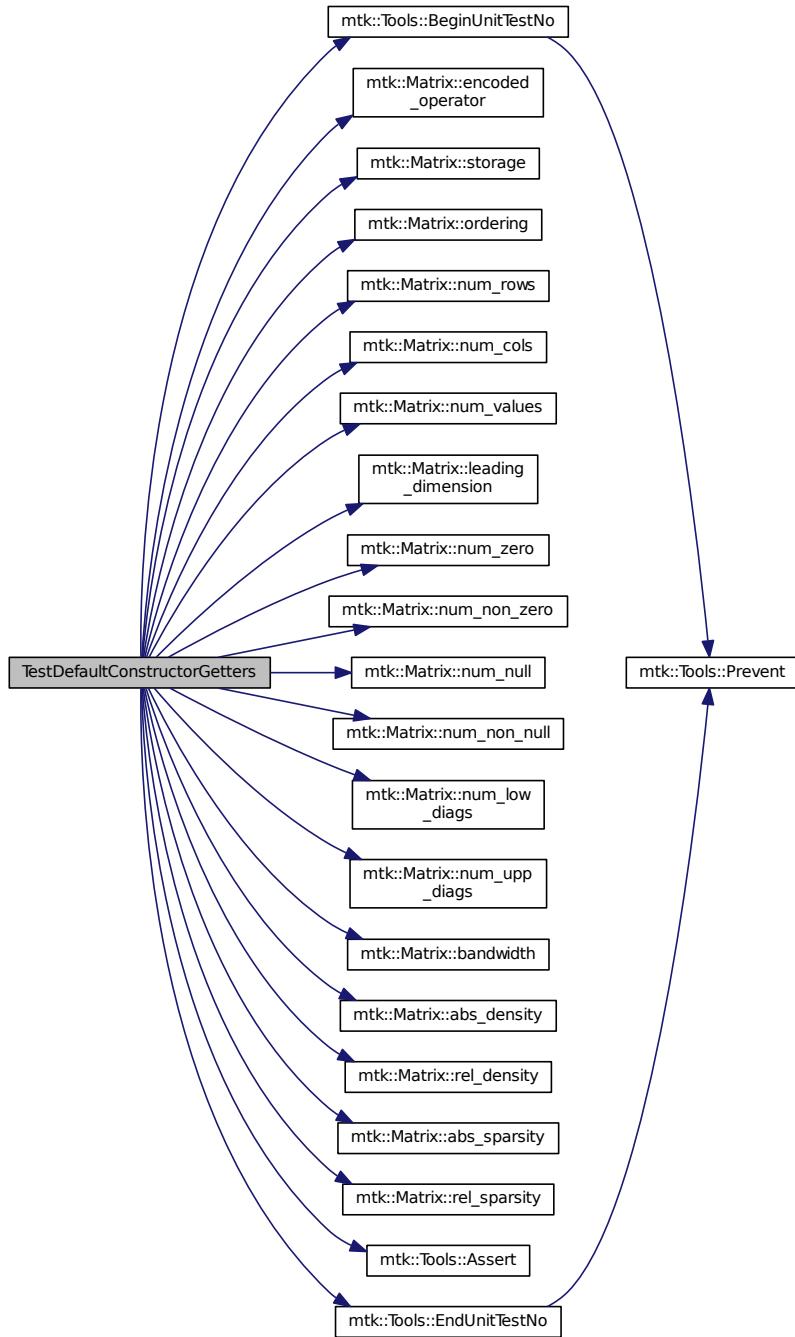
Here is the call graph for this function:



18.163.2.2 void TestDefaultConstructorGetters ()

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

Here is the call graph for this function:



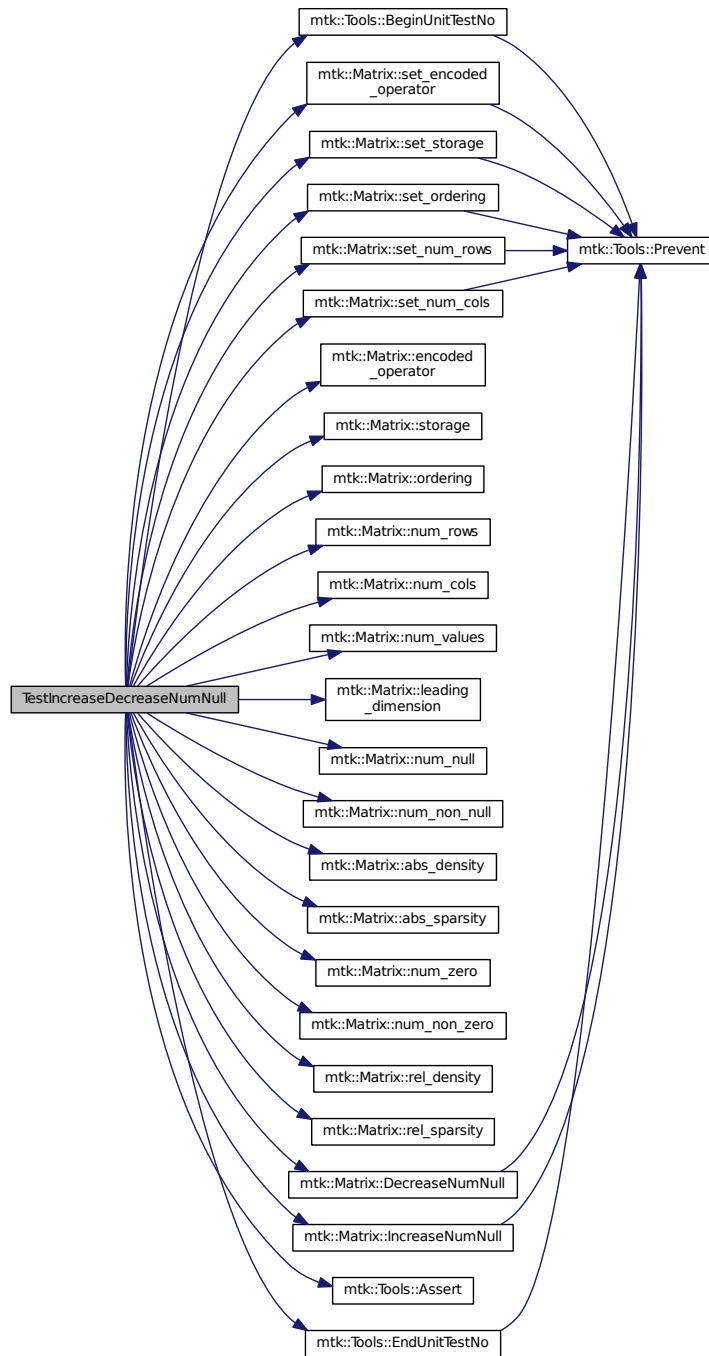
Here is the caller graph for this function:



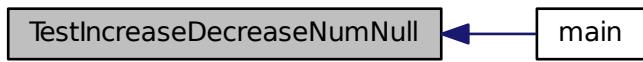
18.163.2.3 void TestIncreaseDecreaseNumNull()

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

Here is the call graph for this function:



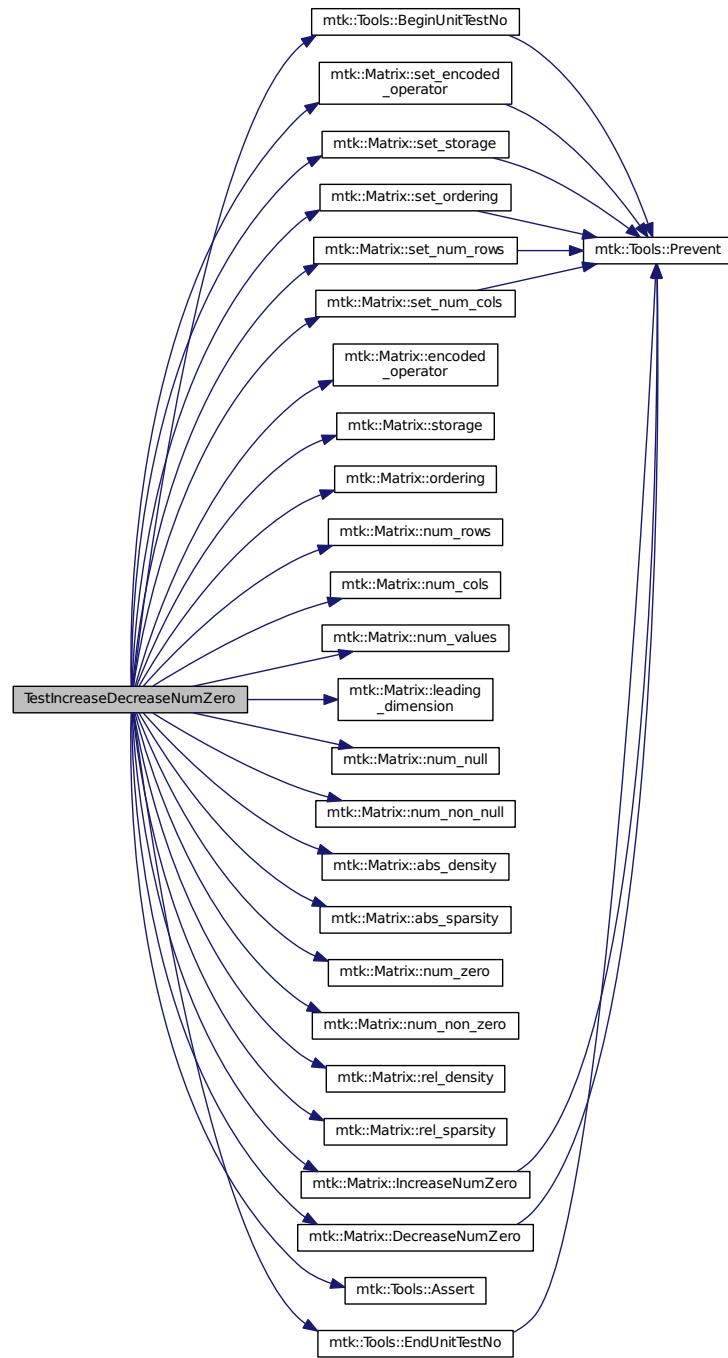
Here is the caller graph for this function:



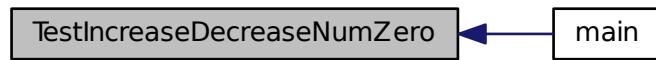
18.163.2.4 void TestIncreaseDecreaseNumZero()

Definition at line 237 of file [mtk_matrix_test.cc](#).

Here is the call graph for this function:



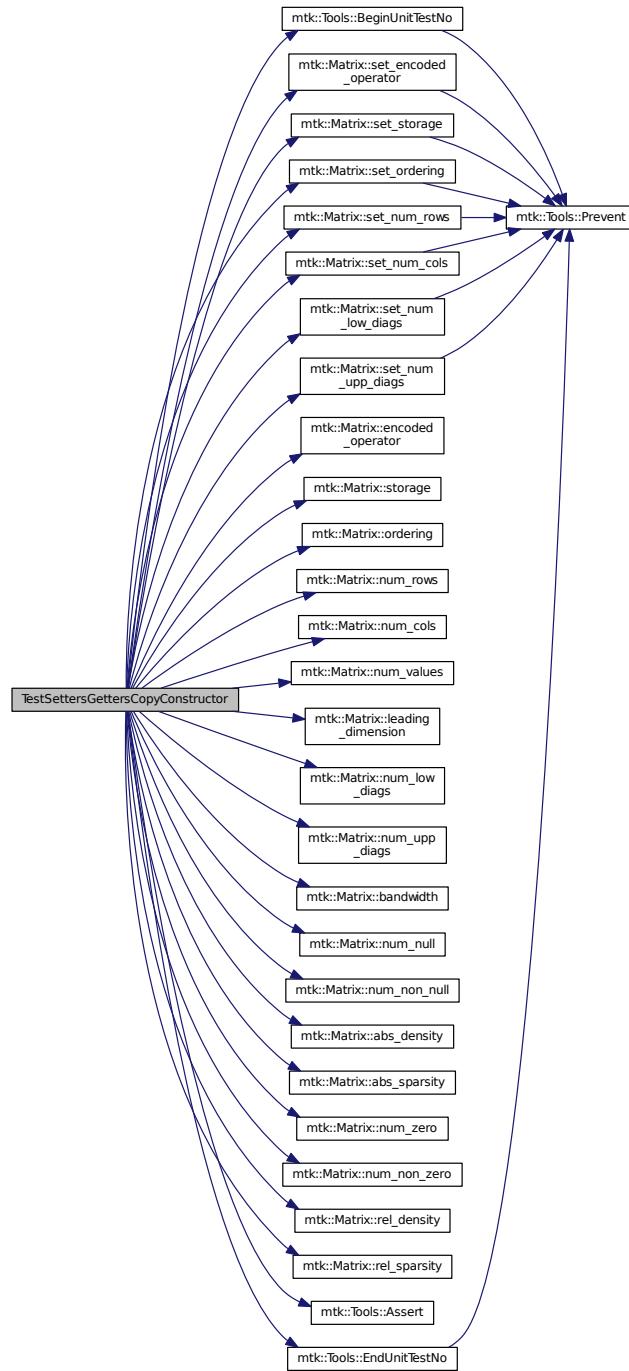
Here is the caller graph for this function:



18.163.2.5 void TestSettersGettersCopyConstructor()

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

Here is the call graph for this function:



Here is the caller graph for this function:



18.164 mtk_matrix_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructorGetters() {
00062     mtk::Tools::BeginUnitTests(1);
  
```

```

00064     mtk::Matrix mm; // A matrix object containing no data at all.
00065
00066     bool assertion{ (mm.encoded_operator() ==
00067         mtk::EncodedOperator::NOOP) &&
00068         (mm.storage() == mtk::MatrixStorage::DENSE) &&
00069         (mm.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00070         (mm.num_rows() == 0) &&
00071         (mm.num_cols() == 0) &&
00072         (mm.num_values() == 0) &&
00073         (mm.leading_dimension() == 0) &&
00074         (mm.num_zero() == 0) &&
00075         (mm.num_non_zero() == 0) &&
00076         (mm.num_null() == 0) &&
00077         (mm.num_non_null() == 0) &&
00078         (mm.num_low_diags() == 0) &&
00079         (mm.num_upp_diags() == 0) &&
00080         (mm.bandwidth() == 0) &&
00081         (mm.abs_density() == mtk::kZero) &&
00082         (mm.rel_density() == mtk::kZero) &&
00083         (mm.abs_sparsity() == mtk::kZero) &&
00084         (mm.rel_sparsity() == mtk::kZero) };
00085
00086     mtk::Tools::Assert(assertion);
00087
00088     mtk::Tools::EndUnitTestNo(1);
00089 }
00090
00091 void TestSettersGettersCopyConstructor() {
00092
00093     mtk::Tools::BeginUnitTestNo(2);
00094
00095     const int num_rows{14};
00096     const int num_cols{17};
00097     const int num_values{num_rows*num_cols};
00098     const int num_low_diags{1};
00099     const int num_upp_diags{1};
00100     const int bandwidth{3};
00101     const int num_non_null{};
00102     const int num_zero{};
00103
00104     mtk::Matrix m1; // A matrix object containing no data at all.
00105
00106     m1.set_encoded_operator(mtk::EncodedOperator::GRADIENT);
00107 ;
00108     m1.set_storage(mtk::MatrixStorage::DENSE);
00109     m1.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00110     m1.set_num_rows(num_rows);
00111     m1.set_num_cols(num_cols);
00112     m1.set_num_low_diags(1);
00113     m1.set_num_upp_diags(1);
00114
00115     bool assertion{ (m1.encoded_operator() ==
00116         mtk::EncodedOperator::GRADIENT) &&
00117         (m1.storage() == mtk::MatrixStorage::DENSE) &&
00118         (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00119         (m1.num_rows() == num_rows) &&
00120         (m1.num_cols() == num_cols) &&
00121         (m1.num_values() == num_values) &&
00122         (m1.leading_dimension() == num_cols) &&
00123         (m1.num_low_diags() == num_low_diags) &&
00124         (m1.num_upp_diags() == num_upp_diags) &&
00125         (m1.bandwidth() == bandwidth) &&
00126         (m1.num_null() == num_values) &&
00127         (m1.num_non_null() == num_non_null) &&
00128         (m1.abs_density() == mtk::kZero) &&
00129         (m1.abs_sparsity() == mtk::kOne) &&
00130         (m1.num_zero() == num_zero) &&
00131         (m1.num_non_zero() == num_values) &&
00132         (m1.rel_density() == mtk::kZero) &&
00133         (m1.rel_sparsity() == mtk::kOne) };
00134
00135     mtk::Matrix m2(m1); // A matrix object that should be a copy of m1.
00136
00137     assertion = assertion &&
00138         (m1.encoded_operator() == m2.encoded_operator()) &&
00139         (m1.storage() == m2.storage()) &&
00140         (m1.ordering() == m2.ordering()) &&
00141         (m1.num_rows() == m2.num_rows()) &&
00142         (m1.num_cols() == m2.num_cols()) &&
00143         (m1.num_values() == m2.num_values()) &&

```

```

00142     (m1.leading_dimension() == m2.leading_dimension()) &&
00143     (m1.num_null() == m2.num_null()) &&
00144     (m1.num_non_null() == m2.num_non_null()) &&
00145     (m1.abs_density() == m2.abs_density()) &&
00146     (m1.abs_sparsity() == m2.abs_sparsity()) &&
00147     (m1.num_zero() == m2.num_zero()) &&
00148     (m1.num_non_zero() == m2.num_non_zero()) &&
00149     (m1.rel_density() == m2.abs_density()) &&
00150     (m1.rel_sparsity() == m2.rel_sparsity());
00151
00152     mtk::Tools::Assert(assertion);
00153
00154     mtk::Tools::EndUnitTestNo(2);
00155 }
00156
00157 void TestIncreaseDecreaseNumNull() {
00158
00159     mtk::Tools::BeginUnitTestNo(3);
00160
00161     const int num_rows{19};
00162     const int num_cols{27};
00163     const int num_values{num_rows*num_cols};
00164     const int num_null{num_values - 1};
00165     const int num_non_null{1};
00166
00167     mtk::Matrix m1; // A matrix object containing no data at all.
00168
00169     mtk::Real abs_density{0.0019493177};
00170     mtk::Real abs_sparsity{0.99805068};
00171
00172     m1.set_encoded_operator(mtk::EncodedOperator::DIVERGENCE)
00173 );
00174     m1.set_storage(mtk::MatrixStorage::DENSE);
00175     m1.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00176     m1.set_num_rows(num_rows);
00177     m1.set_num_cols(num_cols);
00178
00179     bool assertion{(m1.encoded_operator() ==
00180         mtk::EncodedOperator::DIVERGENCE) &&
00181         (m1.storage() == mtk::MatrixStorage::DENSE) &&
00182         (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00183         (m1.num_rows() == num_rows) &&
00184         (m1.num_cols() == num_cols) &&
00185         (m1.num_values() == num_values) &&
00186         (m1.leading_dimension() == num_cols) &&
00187         (m1.num_null() == num_values) &&
00188         (m1.num_non_null() == 0) &&
00189         (m1.abs_density() == mtk::kZero) &&
00190         (m1.abs_sparsity() == mtk::kOne) &&
00191         (m1.num_zero() == 0) &&
00192         (m1.num_non_zero() == num_values) &&
00193         (m1.rel_density() == mtk::kZero) &&
00194         (m1.rel_sparsity() == mtk::kOne));
00195
00196     assertion = assertion &&
00197     (m1.encoded_operator() == mtk::EncodedOperator::DIVERGENCE)
00198 ) &&
00199     (m1.storage() == mtk::MatrixStorage::DENSE) &&
00200     (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00201     (m1.num_rows() == num_rows) &&
00202     (m1.num_cols() == num_cols) &&
00203     (m1.num_values() == num_values) &&
00204     (m1.leading_dimension() == num_cols) &&
00205     (m1.num_null() == num_null) &&
00206     (std::abs(m1.abs_density() - abs_density) <
00207         mtk::kDefaultTolerance) &&
00208     (std::abs(m1.abs_sparsity() - abs_sparsity) <
00209         mtk::kDefaultTolerance) &&
00210     (m1.num_zero() == 0) &&
00211     (m1.num_non_zero() == num_values) &&
00212     (m1.rel_density() == mtk::kZero) &&
00213     (m1.rel_sparsity() == mtk::kOne);
00214
00215     m1.IncreaseNumNull();
00216
00217     assertion = assertion &&
00218     (m1.encoded_operator() == mtk::EncodedOperator::DIVERGENCE)
00219 ) &&

```

```

00217     (m1.storage() == mtk::MatrixStorage::DENSE) &&
00218     (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00219     (m1.num_rows() == num_rows) &&
00220     (m1.num_cols() == num_cols) &&
00221     (m1.num_values() == num_values) &&
00222     (m1.leading_dimension() == num_cols) &&
00223     (m1.num_null() == num_values) &&
00224     (m1.num_non_null() == 0) &&
00225     (m1.abs_density() == mtk::kZero) &&
00226     (m1.abs_sparsity() == mtk::kOne) &&
00227     (m1.num_zero() == 0) &&
00228     (m1.num_non_zero() == num_values) &&
00229     (m1.rel_density() == mtk::kZero) &&
00230     (m1.rel_sparsity() == mtk::kOne);
00231
00232     mtk::Tools::Assert(assertion);
00233
00234     mtk::Tools::EndUnitTestNo(3);
00235 }
00236
00237 void TestIncreaseDecreaseNumZero() {
00238
00239     mtk::Tools::BeginUnitTestNo(4);
00240
00241     const int num_rows{19};
00242     const int num_cols{27};
00243     const int num_values{num_rows*num_cols};
00244
00245     mtk::Matrix m1; // A matrix object containing no data at all.
00246
00247     mtk::Real abs_density{0.0019493177};
00248     mtk::Real abs_sparsity{0.99805068};
00249     mtk::Real rel_density{0.0019493177};
00250     mtk::Real rel_sparsity{0.99805068};
00251
00252     m1.set_encoded_operator(mtk::EncodedOperator::DIVERGENCE
00253 );
00254     m1.set_storage(mtk::MatrixStorage::DENSE);
00255     m1.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00256     m1.set_num_rows(19);
00257     m1.set_num_cols(27);
00258
00259     bool assertion{ (m1.encoded_operator() ==
00260         mtk::EncodedOperator::DIVERGENCE) &&
00261         (m1.storage() == mtk::MatrixStorage::DENSE) &&
00262         (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00263         (m1.num_rows() == num_rows) &&
00264         (m1.num_cols() == num_cols) &&
00265         (m1.num_values() == num_values) &&
00266         (m1.leading_dimension() == num_cols) &&
00267         (m1.num_null() == num_values) &&
00268         (m1.num_non_null() == 0) &&
00269         (m1.abs_density() == mtk::kZero) &&
00270         (m1.abs_sparsity() == mtk::kOne) &&
00271         (m1.num_zero() == 0) &&
00272         (m1.num_non_zero() == num_values) &&
00273         (m1.rel_density() == mtk::kZero) &&
00274         (m1.rel_sparsity() == mtk::kOne) };
00275
00276     m1.IncreaseNumZero();
00277
00278     assertion = assertion &&
00279         (m1.encoded_operator() == mtk::EncodedOperator::DIVERGENCE
00280 ) &&
00281         (m1.storage() == mtk::MatrixStorage::DENSE) &&
00282         (m1.ordering() == mtk::MatrixOrdering::ROW_MAJOR) &&
00283         (m1.num_rows() == num_rows) &&
00284         (m1.num_cols() == num_cols) &&
00285         (m1.num_values() == num_values) &&
00286         (m1.leading_dimension() == num_cols) &&
00287         (m1.num_null() == num_values - 1) &&
00288         (m1.num_non_null() == 1) &&
00289         (std::abs(m1.abs_density() - abs_density) <
00290             mtk::kDefaultTolerance) &&
00291         (std::abs(m1.abs_sparsity() - abs_sparsity) <
00292             mtk::kDefaultTolerance) &&
00293         (m1.num_zero() == 1) &&
00294         (m1.num_non_zero() == num_values - 1) &&
00295         (std::abs(m1.rel_density() - rel_density) <
00296             mtk::kDefaultTolerance) &&
00297         (std::abs(m1.rel_sparsity() - rel_sparsity) <

```

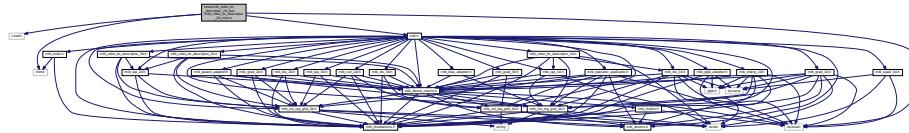
```

00292     mtk::kDefaultTolerance);
00293     m1.DecreaseNumZero();
00294
00295     mtk::Tools::Assert(assertion);
00296
00297     mtk::Tools::EndUnitTestNo(4);
00298 }
00299
00300 int main (int argc, char *argv[]) {
00301
00302     std::cout << "Testing mtk::Matrix class." << std::endl;
00303
00304     TestDefaultConstructorGetters();
00305     TestSettersGettersCopyConstructor();
00306     TestIncreaseDecreaseNumNull();
00307     TestIncreaseDecreaseNumZero();
00308 }
```

18.165 tests/mtk_robin_bc_descriptor_2d_test/mtk_robin_bc_descriptor_2d_test.cc File Reference

Test file for the [mtk::RobinBCDescriptor2D](#) class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_robin_bc_descriptor_2d_test.cc:
```



Functions

- void [TestDefaultConstructorGetters \(\)](#)
- [mtk::Real cc \(const mtk::Real &xx, const mtk::Real &yy\)](#)
- void [TestPushBackImposeOnLaplacianMatrix \(\)](#)
- [mtk::Real ScalarField \(const mtk::Real &xx, const mtk::Real &yy\)](#)
- [mtk::Real HomogeneousDiricheletBC \(const mtk::Real &xx, const mtk::Real &tt\)](#)
- void [TestImposeOnGrid \(\)](#)
- int [main \(\)](#)

18.165.1 Detailed Description

Author

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

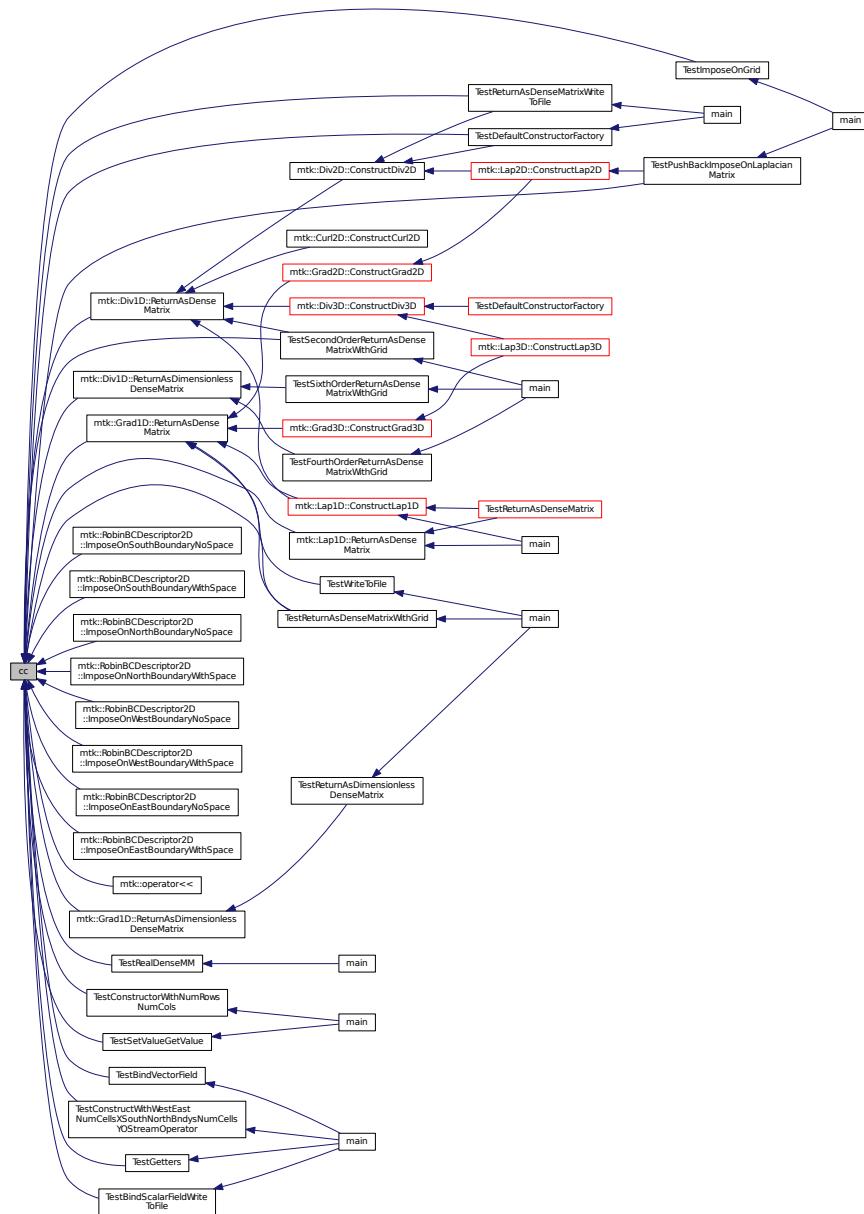
Definition in file [mtk_robin_bc_descriptor_2d_test.cc](#).

18.165.2 Function Documentation

18.165.2.1 mtk::Real cc (const mtk::Real & xx, const mtk::Real & yy)

Definition at line 78 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

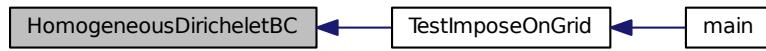
Here is the caller graph for this function:



18.165.2.2 mtk::Real HomogeneousDiricheletBC (const mtk::Real & xx, const mtk::Real & tt)

Definition at line 139 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

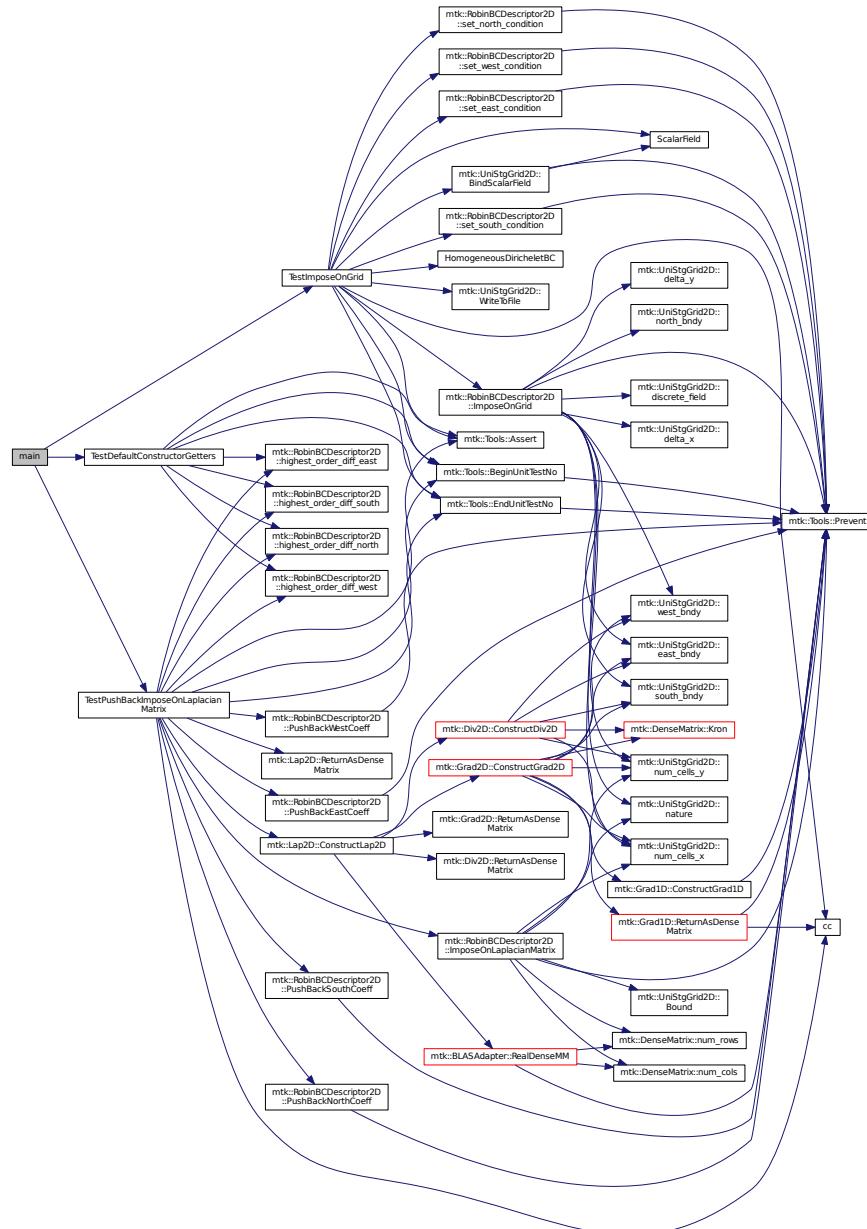
Here is the caller graph for this function:



18.165.2.3 int main ()

Definition at line 183 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

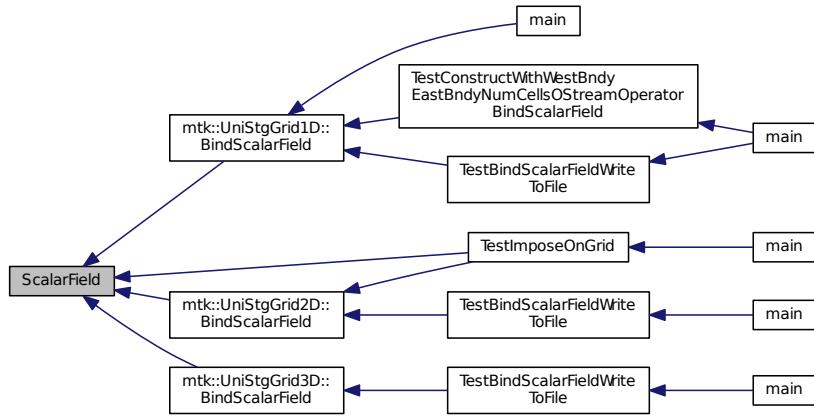
Here is the call graph for this function:



18.165.2.4 mtk::Real ScalarField (const mtk::Real & xx, const mtk::Real & yy)

Definition at line 132 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

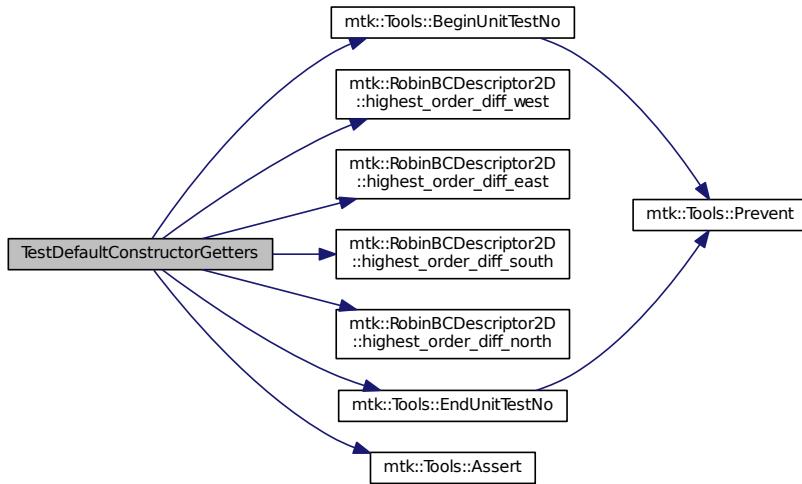
Here is the caller graph for this function:



18.165.2.5 void TestDefaultConstructorGetters ()

Definition at line 61 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

Here is the call graph for this function:



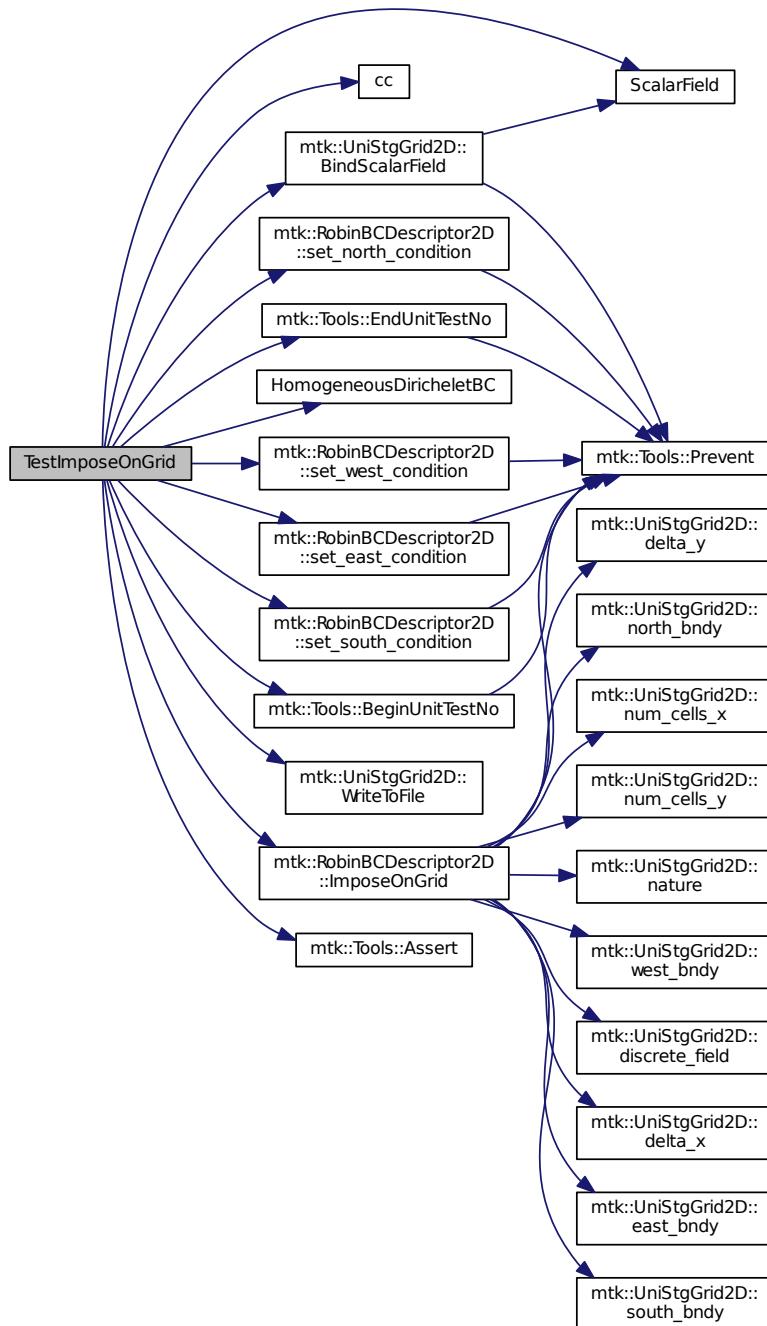
Here is the caller graph for this function:



18.165.2.6 void TestImposeOnGrid()

Definition at line 145 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

Here is the call graph for this function:



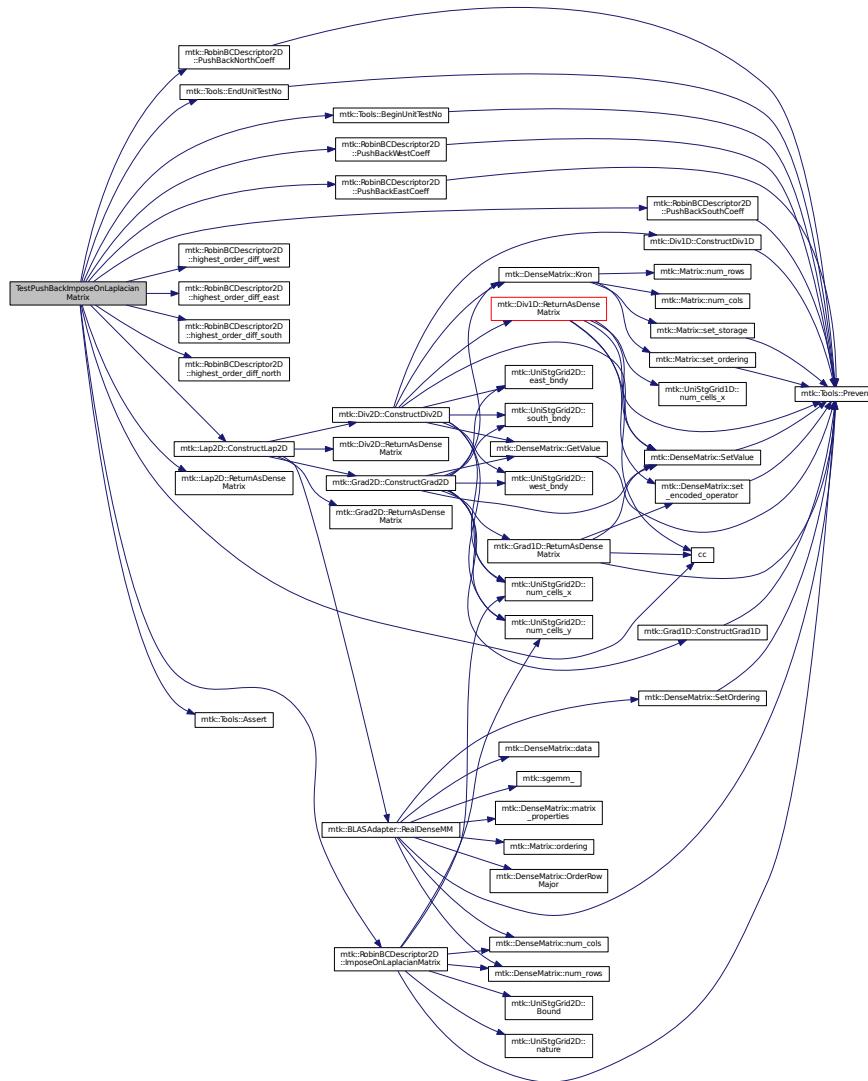
Here is the caller graph for this function:



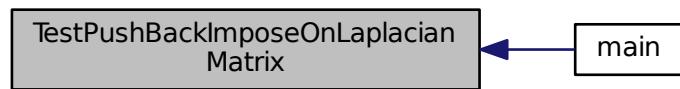
18.165.2.7 void TestPushBackImposeOnLaplacianMatrix()

Definition at line 83 of file [mtk_robin_bc_descriptor_2d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.166 mtk_robin_bc_descriptor_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
0010 University. All rights reserved.
0011
0012 Redistribution and use in source and binary forms, with or without modification,
0013 are permitted provided that the following conditions are met:
0014
0015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
0016 and a copy of the modified files should be reported once modifications are
0017 completed, unless these modifications are made through the project's GitHub
0018 page: http://www.csric.sdsu.edu/mtk. Documentation related to said modifications
0019 should be developed and included in any deliverable.
0020
0021 2. Redistributions of source code must be done through direct
0022 downloads from the project's GitHub page: http://www.csric.sdsu.edu/mtk
0023
0024 3. Redistributions in binary form must reproduce the above copyright notice,
0025 this list of conditions and the following disclaimer in the documentation and/or
0026 other materials provided with the distribution.
0027
0028 4. Usage of the binary form on proprietary applications shall require explicit
0029 prior written permission from the the copyright holders, and due credit should
0030 be given to the copyright holders.
0031
0032 5. Neither the name of the copyright holder nor the names of its contributors
0033 may be used to endorse or promote products derived from this software without
0034 specific prior written permission.
0035
0036 The copyright holders provide no reassurances that the source code provided does
0037 not infringe any patent, copyright, or any other intellectual property rights of
0038 third parties. The copyright holders disclaim any liability to any recipient for
0039 claims brought against recipient by any third party for infringement of that
0040 parties intellectual property rights.
0041
0042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
0043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
0044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
0045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
0046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
0047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
0048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
0049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
0050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
0051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
0052 */
0053
0054 #include <cmath>
0055 #include <ctime>
0056
0057 #include <iostream>
0058
0059 #include "mtk.h"
0060
0061 void TestDefaultConstructorGetters() {
0062
0063     mtk::Tools::BeginUnitTestNo(1);
0064
0065     mtk::RobinBCDescriptor2D bcd;
0066
0067     bool assertion{true};
0068
0069     assertion = assertion && bcd.highest_order_diff_west() == -1;
0070     assertion = assertion && bcd.highest_order_diff_east() == -1;
0071     assertion = assertion && bcd.highest_order_diff_south() == -1;
0072     assertion = assertion && bcd.highest_order_diff_north() == -1;
0073
0074     mtk::Tools::EndUnitTestNo(1);
0075     mtk::Tools::Assert(assertion);
0076 }
0077
0078 mtk::Real cc(const mtk::Real &xx, const mtk::Real &yy) {
0079
0080     return mtk::kOne;
0081 }
0082
0083 void TestPushBackImposeOnLaplacianMatrix() {
0084 }
```

```

00085     mtk::Tools::BeginUnitTestNo(2);
00086
00087     mtk::RobinBCDescriptor2D bcd;
00088
00089     bool assertion{true};
00090
00091     bcd.PushBackWestCoeff(cc);
00092     bcd.PushBackEastCoeff(cc);
00093     bcd.PushBackSouthCoeff(cc);
00094     bcd.PushBackNorthCoeff(cc);
00095
00096     assertion = assertion && bcd.highest_order_diff_west() == 0;
00097     assertion = assertion && bcd.highest_order_diff_east() == 0;
00098     assertion = assertion && bcd.highest_order_diff_south() == 0;
00099     assertion = assertion && bcd.highest_order_diff_north() == 0;
00100
00101     mtk::Real aa = 0.0;
00102     mtk::Real bb = 1.0;
00103     mtk::Real cc = 0.0;
00104     mtk::Real dd = 1.0;
00105
00106     int nn = 5;
00107     int mm = 5;
00108
00109     mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00110
00111     mtk::Lap2D ll;
00112
00113     assertion = ll.ConstructLap2D(llg);
00114
00115     if (!assertion) {
00116         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00117     }
00118
00119     mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00120
00121     assertion = assertion && (llm.num_rows() != 0);
00122
00123     bcd.ImposeOnLaplacianMatrix(ll, llg, llm);
00124
00125     assertion = assertion &&
00126         llm.WriteToFile("mtk_robin_bc_descriptor_2d_test_02.dat");
00127
00128     mtk::Tools::EndUnitTestNo(2);
00129     mtk::Tools::Assert(assertion);
00130 }
00131
00132 mtk::Real ScalarField(const mtk::Real &xx, const
00133     mtk::Real &yy) {
00134
00135     mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
00136
00137     return xx*yy*exp(aux);
00138 }
00139 mtk::Real HomogeneousDiricheletBC(const
00140     mtk::Real &xx,
00141                                     const mtk::Real &tt) {
00142
00143     return mtk::kZero;
00144 }
00145 void TestImposeOnGrid() {
00146
00147     mtk::Tools::BeginUnitTestNo(3);
00148
00149     mtk::Real aa = 0.0;
00150     mtk::Real bb = 1.0;
00151     mtk::Real cc = 0.0;
00152     mtk::Real dd = 1.0;
00153
00154     int nn = 5;
00155     int mm = 5;
00156
00157     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00158
00159     gg.BindScalarField(ScalarField);
00160
00161     mtk::RobinBCDescriptor2D desc;
00162
00163     desc.set_west_condition(HomogeneousDiricheletBC);

```

```

00164     desc.set_east_condition(HomogeneousDiricheletBC);
00165     desc.set_south_condition(HomogeneousDiricheletBC);
00166     desc.set_north_condition(HomogeneousDiricheletBC);
00167
00168     desc.ImposeOnGrid(gg);
00169
00170     bool assertion(gg.WriteLine("mtk_robin_bc_descriptor_2d_test_03.dat",
00171                     "x",
00172                     "y",
00173                     "u(x,y)"));
00174
00175     if(!assertion) {
00176         std::cerr << "Error writing to file." << std::endl;
00177     }
00178
00179     mtk::Tools::EndUnitTestNo(3);
00180     mtk::Tools::Assert(assertion);
00181 }
00182
00183 int main () {
00184
00185     std::cout << "Testing mtk::RobinBCDescriptor2D class." << std::endl;
00186
00187     TestDefaultConstructorGetters();
00188     TestPushBackImposeOnLaplacianMatrix();
00189     TestImposeOnGrid();
00190 }
```

18.167 tests/mtk_uni_stg_grid_1d_test/mtk_uni_stg_grid_1d_test.cc File Reference

Test file for the [mtk::UniStgGrid1D](#) class.

```
#include <ctime>
#include <iostream>
#include <vector>
#include "mtk.h"
Include dependency graph for mtk_uni_stg_grid_1d_test.cc:
```



Functions

- void [TestDefaultConstructor](#) ()
- [mtk::Real](#) [ScalarField](#) (const [mtk::Real](#) &xx, const std::vector<[mtk::Real](#)> &pp)
- void [TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField](#) ()
- void [TestBindScalarFieldWriteToFile](#) ()
- [mtk::Real](#) [VectorFieldPComponent](#) (const [mtk::Real](#) &xx, const std::vector<[mtk::Real](#)> &pp)
- void [TestBindVectorField](#) ()
- int [main](#) ()

18.167.1 Detailed Description

Author

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

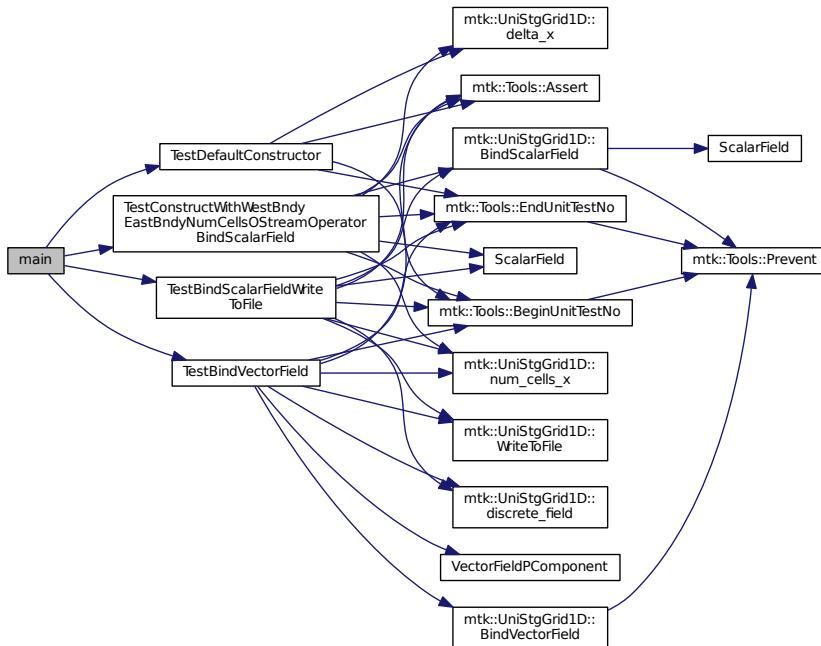
Definition in file [mtk_uni_stg_grid_1d_test.cc](#).

18.167.2 Function Documentation

18.167.2.1 int main()

Definition at line 160 of file [mtk_uni_stg_grid_1d_test.cc](#).

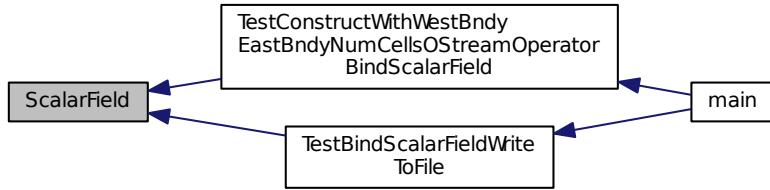
Here is the call graph for this function:



18.167.2.2 mtk::Real ScalarField (const mtk::Real & xx, const std::vector< mtk::Real > & pp)

Definition at line 72 of file [mtk_uni_stg_grid_1d_test.cc](#).

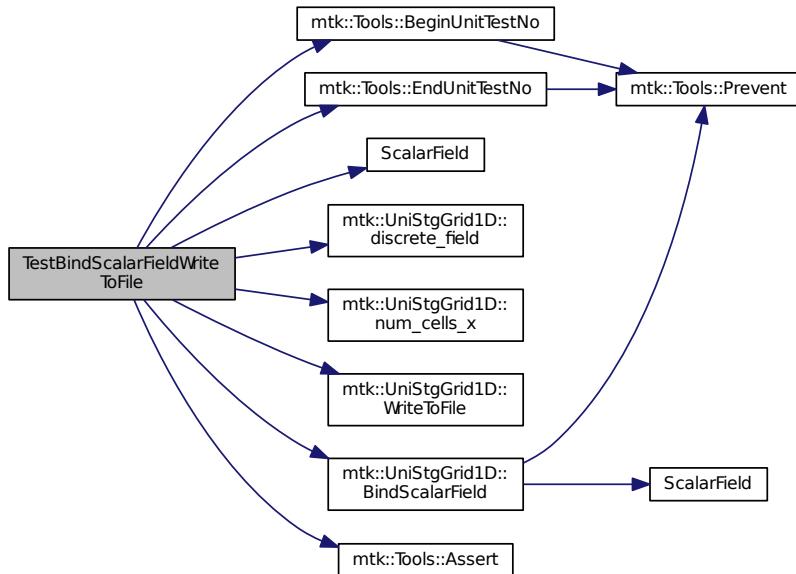
Here is the caller graph for this function:



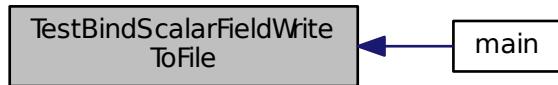
18.167.2.3 void TestBindScalarFieldWriteToFile()

Definition at line 96 of file [mtk_uni_stg_grid_1d_test.cc](#).

Here is the call graph for this function:



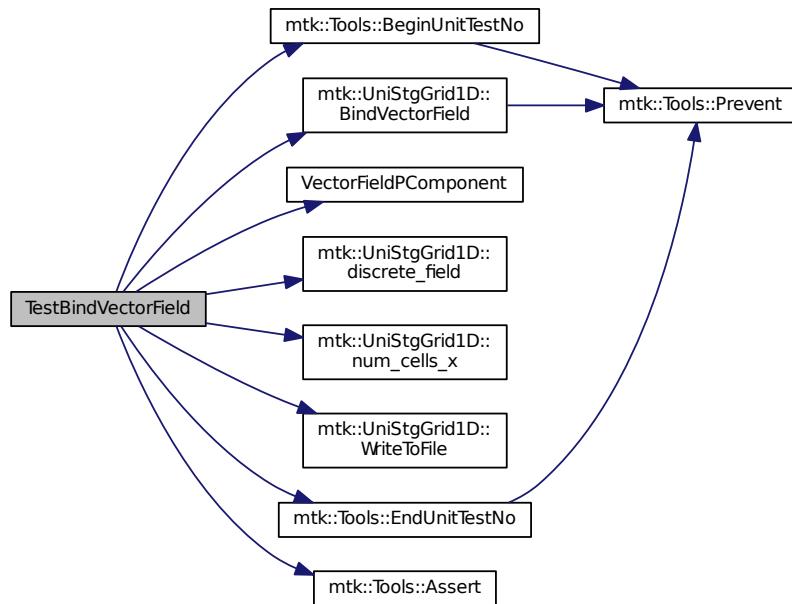
Here is the caller graph for this function:



18.167.2.4 void TestBindVectorField()

Definition at line 131 of file [mtk_uni_stg_grid_1d_test.cc](#).

Here is the call graph for this function:



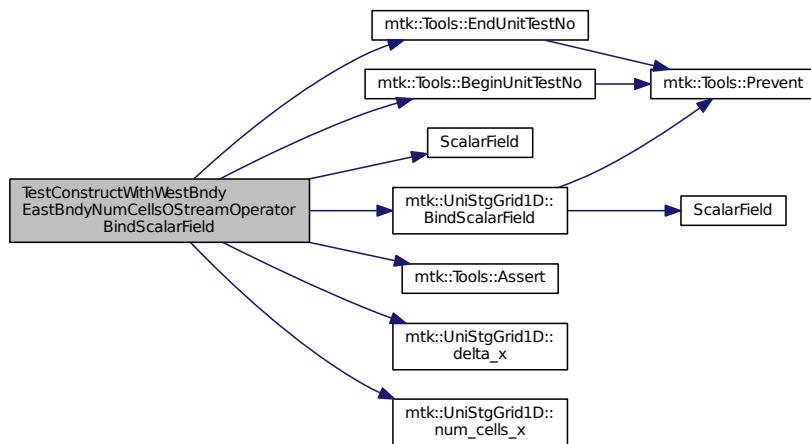
Here is the caller graph for this function:



18.167.2.5 void TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField()

Definition at line 77 of file [mtk_uni_stg_grid_1d_test.cc](#).

Here is the call graph for this function:



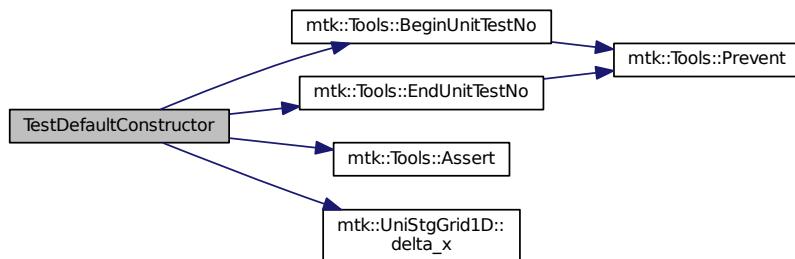
Here is the caller graph for this function:



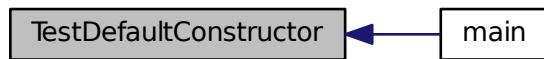
18.167.2.6 void TestDefaultConstructor()

Definition at line 62 of file [mtk_uni_stg_grid_1d_test.cc](#).

Here is the call graph for this function:



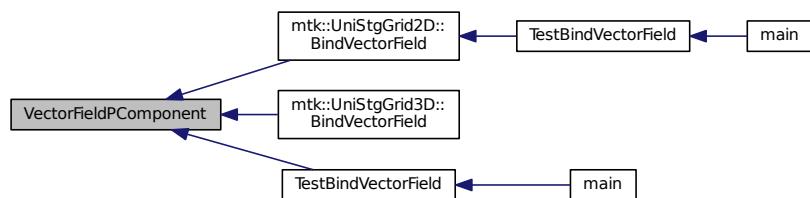
Here is the caller graph for this function:



18.167.2.7 mtk::Real VectorFieldPComponent(const mtk::Real & xx, const std::vector<mtk::Real> & pp)

Definition at line 125 of file [mtk_uni_stg_grid_1d_test.cc](#).

Here is the caller graph for this function:



18.168 mtk_uni_stg_grid_1d_test.cc

```

00001
00008 

```

```

00084     int nn = 5;
00085
00086     mtk::UniStgGrid1D gg(aa, bb, nn);
00087
00088     gg.BindScalarField(ScalarField, std::vector<mtk::Real>());
00089
00090     std::cout << gg << std::endl;
00091
00092     mtk::Tools::EndUnitTestNo(2);
00093     mtk::Tools::Assert(gg.delta_x() == 0.2 && gg.
00094     num_cells_x() == 5);
00094 }
00095
00096 void TestBindScalarFieldWriteToFile() {
00097
00098     mtk::Tools::BeginUnitTestNo(3);
00099
00100     mtk::Real aa = 0.0;
00101     mtk::Real bb = 1.0;
00102
00103     int nn = 5;
00104
00105     mtk::UniStgGrid1D gg(aa, bb, nn);
00106
00107     bool assertion{true};
00108
00109     gg.BindScalarField(ScalarField, std::vector<mtk::Real>());
00110
00111     assertion =
00112         assertion &&
00113         gg.discrete_field()[0] == 0.0 &&
00114         gg.discrete_field()[gg.num_cells_x() + 2 - 1] == 2.0;
00115
00116     if(!gg.WriteLine("mtk_uni_stg_grid_1d_test_03_gg.dat", "x", "u(x)")) {
00117         std::cerr << "Error writing to file." << std::endl;
00118         assertion = false;
00119     }
00120
00121     mtk::Tools::EndUnitTestNo(3);
00122     mtk::Tools::Assert(assertion);
00123 }
00124
00125 mtk::Real VectorFieldPComponent(const mtk::Real &xx,
00126                                     const std::vector<mtk::Real> &pp) {
00127
00128     return xxxx;
00129 }
00130
00131 void TestBindVectorField() {
00132
00133     mtk::Tools::BeginUnitTestNo(4);
00134
00135     mtk::Real aa = 0.0;
00136     mtk::Real bb = 1.0;
00137
00138     int nn = 20;
00139
00140     mtk::UniStgGrid1D gg(aa, bb, nn, mtk::FieldNature::VECTOR);
00141
00142     bool assertion{true};
00143
00144     gg.BindVectorField(VectorFieldPComponent, std::vector<mtk::Real>());
00145
00146     assertion =
00147         assertion &&
00148         gg.discrete_field()[0] == 0.0 &&
00149         gg.discrete_field()[gg.num_cells_x() + 1 - 1] == 1.0;
00150
00151     if(!gg.WriteLine("mtk_uni_stg_grid_1d_test_04_gg.dat", "x", "v(x)")) {
00152         std::cerr << "Error writing to file." << std::endl;
00153         assertion = false;
00154     }
00155
00156     mtk::Tools::EndUnitTestNo(4);
00157     mtk::Tools::Assert(assertion);
00158 }
00159
00160 int main () {
00161
00162     std::cout << "Testing mtk::UniStgGrid1D class." << std::endl;
00163

```

```

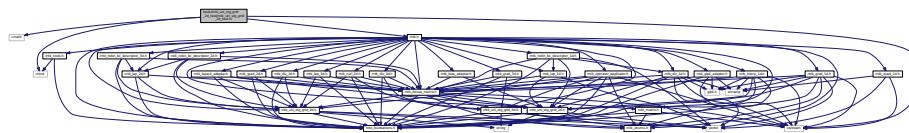
00164     TestDefaultConstructor();
00165     TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField
());
00166     TestBindScalarFieldWriteToFile();
00167     TestBindVectorField();
00168 }

```

18.169 tests/mtk_uni_stg_grid_2d_test/mtk_uni_stg_grid_2d_test.cc File Reference

Test file for the [mtk::UniStgGrid2D](#) class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_uni_stg_grid_2d_test.cc:
```



Functions

- void [TestDefaultConstructor \(\)](#)
- void [TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSStreamOperator \(\)](#)
- void [TestGetters \(\)](#)
- [mtk::Real ScalarField \(const mtk::Real &xx, const mtk::Real &yy\)](#)
- void [TestBindScalarFieldWriteToFile \(\)](#)
- [mtk::Real VectorFieldPComponent \(const mtk::Real &xx, const mtk::Real &yy\)](#)
- [mtk::Real VectorFieldQComponent \(const mtk::Real &xx, const mtk::Real &yy\)](#)
- void [TestBindVectorField \(\)](#)
- int [main \(\)](#)

18.169.1 Detailed Description

Author

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

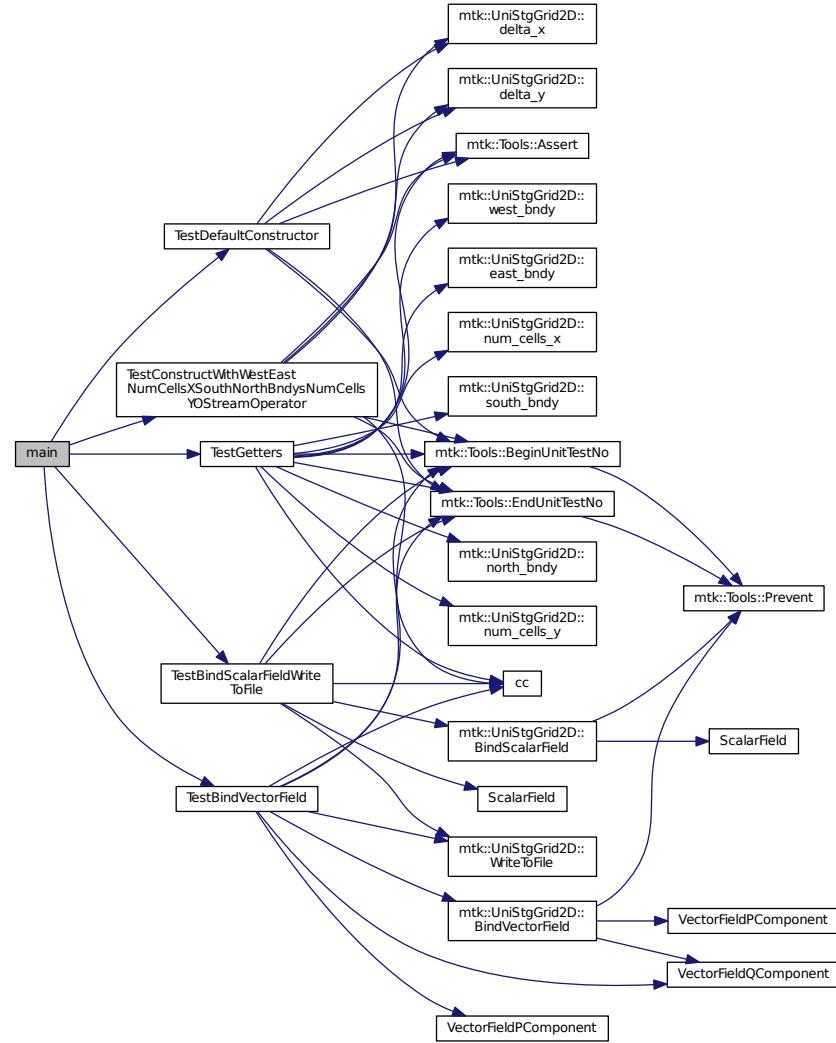
Definition in file [mtk_uni_stg_grid_2d_test.cc](#).

18.169.2 Function Documentation

18.169.2.1 int main ()

Definition at line 185 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



18.169.2.2 mtk::Real ScalarField (const mtk::Real & xx, const mtk::Real & yy)

Definition at line 120 of file [mtk_uni_stg_grid_2d_test.cc](#).

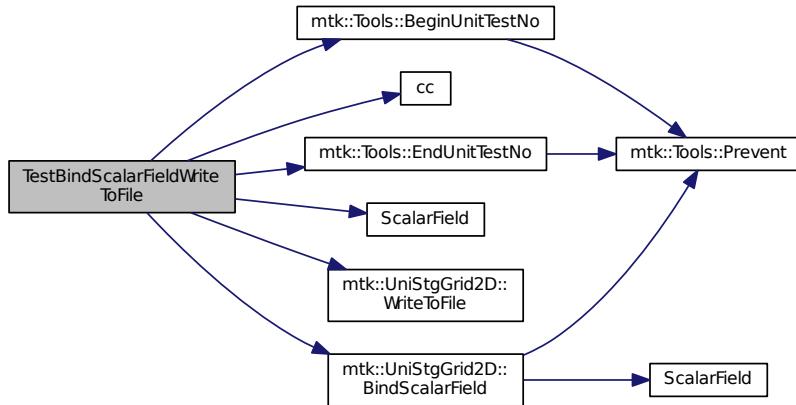
Here is the caller graph for this function:



18.169.2.3 void TestBindScalarFieldWriteToFile()

Definition at line 127 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



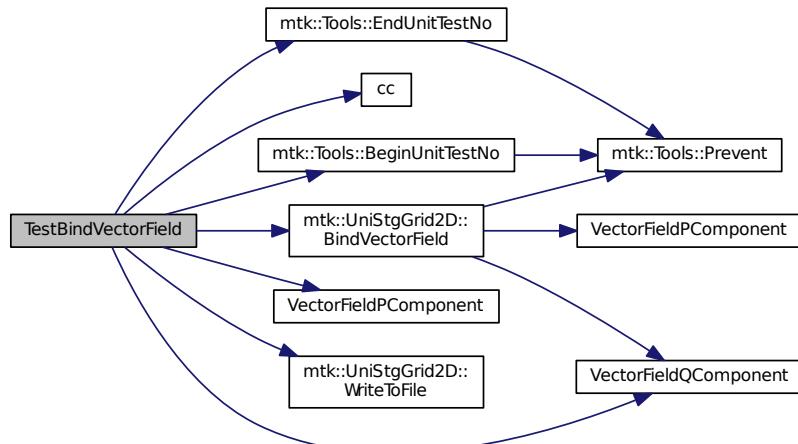
Here is the caller graph for this function:



18.169.2.4 void TestBindVectorField ()

Definition at line 160 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



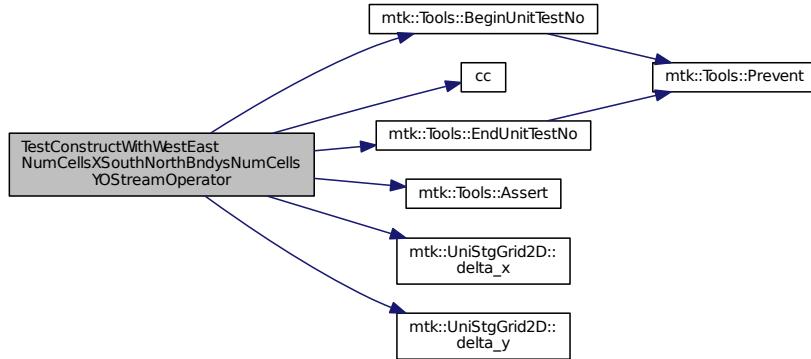
Here is the caller graph for this function:



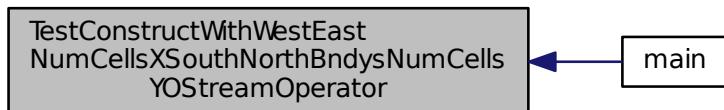
18.169.2.5 void TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSStreamOperator ()

Definition at line 72 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



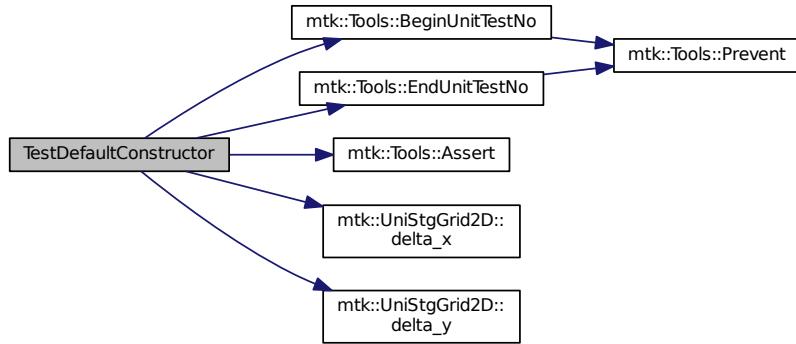
Here is the caller graph for this function:



18.169.2.6 void TestDefaultConstructor()

Definition at line 61 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



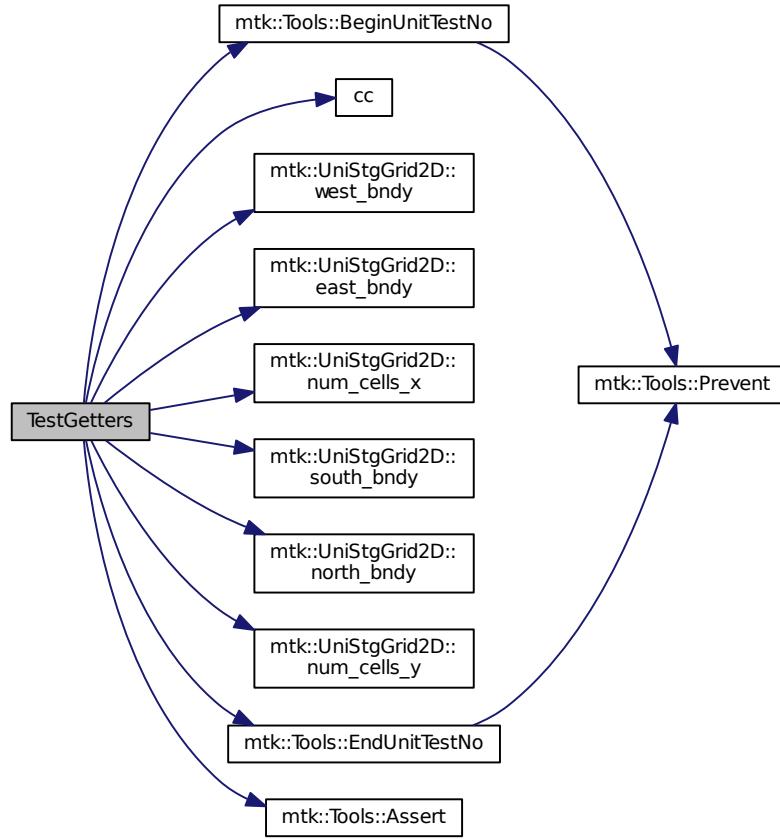
Here is the caller graph for this function:



18.169.2.7 void TestGetters()

Definition at line 93 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the call graph for this function:



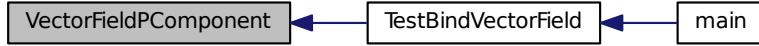
Here is the caller graph for this function:



18.169.2.8 mtk::Real VectorFieldPComponent (const mtk::Real & xx, const mtk::Real & yy)

Definition at line 150 of file [mtk_uni_stg_grid_2d_test.cc](#).

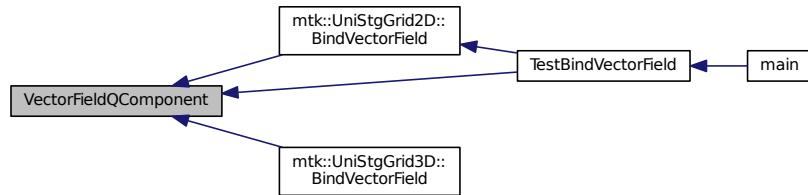
Here is the caller graph for this function:



18.169.2.9 mtk::Real VectorFieldQComponent (const mtk::Real & xx, const mtk::Real & yy)

Definition at line 155 of file [mtk_uni_stg_grid_2d_test.cc](#).

Here is the caller graph for this function:



18.170 mtk_uni_stg_grid_2d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csirc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csirc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of

```

```

00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::UniStgGrid2D gg;
00066
00067     mtk::Tools::EndUnitTestNo(1);
00068     mtk::Tools::Assert(gg.delta_x() == mtk::kZero && gg.
00069     delta_y() == mtk::kZero);
00070 }
00071 void
00072 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator
00073     () {
00074
00075     mtk::Tools::BeginUnitTestNo(2);
00076
00077     mtk::Real aa = 0.0;
00078     mtk::Real bb = 1.0;
00079     mtk::Real cc = 0.0;
00080     mtk::Real dd = 1.0;
00081
00082     int nn = 5;
00083     int mm = 7;
00084
00085     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00086
00087     std::cout << gg << std::endl;
00088
00089     mtk::Tools::EndUnitTestNo(2);
00090     mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00091             abs(gg.delta_y() - 0.142857) <
00092             mtk::kDefaultTolerance);
00093 }
00094 void TestGetters() {
00095
00096     mtk::Tools::BeginUnitTestNo(3);
00097
00098     mtk::Real aa = 0.0;
00099     mtk::Real bb = 1.0;
00100     mtk::Real cc = 0.0;
00101
00102     int nn = 5;
00103     int mm = 7;
00104
00105     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00106
00107     bool assertion{true};
00108
00109     assertion = assertion && (gg.west_bndy() == aa);
00110     assertion = assertion && (gg.east_bndy() == bb);
00111     assertion = assertion && (gg.num_cells_x() == nn);
00112     assertion = assertion && (gg.south_bndy() == cc);
00113     assertion = assertion && (gg.north_bndy() == dd);
00114     assertion = assertion && (gg.num_cells_y() == mm);
00115

```

```

0016     mtk::Tools::EndUnitTestNo(3);
0017     mtk::Tools::Assert(assertion);
0018 }
0019
0020 mtk::Real ScalarField(const mtk::Real &xx, const
0021     mtk::Real &yy) {
0022     mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
0023
0024     return xx*yy*exp(aux);
0025 }
0026
0027 void TestBindScalarFieldWriteToFile() {
0028
0029     mtk::Tools::BeginUnitTestNo(4);
0030
0031     mtk::Real aa = 0.0;
0032     mtk::Real bb = 1.0;
0033     mtk::Real cc = 0.0;
0034     mtk::Real dd = 1.0;
0035
0036     int nn = 5;
0037     int mm = 5;
0038
0039     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
0040
0041     gg.BindScalarField(ScalarField);
0042
0043     if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_04_gg.dat", "x", "y", "u(x,y)")) {
0044         std::cerr << "Error writing to file." << std::endl;
0045     }
0046
0047     mtk::Tools::EndUnitTestNo(4);
0048 }
0049
0050 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const
0051     mtk::Real &yy) {
0052
0053     return xx + 0.01;
0054 }
0055
0056 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
0057     mtk::Real &yy) {
0058
0059     return yy + 0.01;
0060 }
0061
0062 void TestBindVectorField() {
0063
0064     mtk::Tools::BeginUnitTestNo(5);
0065
0066     mtk::Real aa = 0.0;
0067     mtk::Real bb = 1.0;
0068     mtk::Real cc = 0.0;
0069     mtk::Real dd = 1.0;
0070
0071     int nn = 5;
0072     int mm = 5;
0073
0074     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm,
0075         mtk::FieldNature::VECTOR);
0076
0077     gg.BindVectorField(VectorFieldPComponent,
0078         VectorFieldQComponent);
0079
0080     std::cout << gg << std::endl;
0081
0082     if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_05_gg.dat", "x", "y", "v(x,y)")) {
0083         std::cerr << "Error writing to file." << std::endl;
0084     }
0085
0086     mtk::Tools::EndUnitTestNo(5);
0087 }
0088
0089 int main () {
0090
0091     std::cout << "Testing mtk::UniStgGrid2D class." << std::endl;
0092
0093     TestDefaultConstructor();
0094     TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSStreamOperator
0095     ();

```

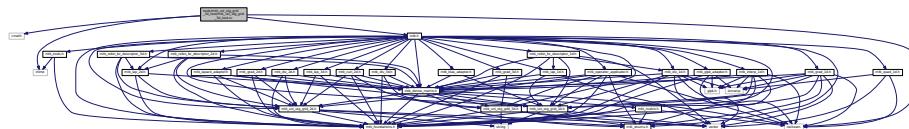
```

00191     TestGetters();
00192     TestBindScalarFieldWriteToFile();
00193     TestBindVectorField();
00194 }
```

18.171 tests/mtk_uni_stg_grid_3d_test/mtk_uni_stg_grid_3d_test.cc File Reference

Test file for the [mtk::UniStgGrid3D](#) class.

```
#include <cmath>
#include <ctime>
#include <iostream>
#include "mtk.h"
Include dependency graph for mtk_uni_stg_grid_3d_test.cc:
```



Functions

- void [TestDefaultConstructor \(\)](#)
- void [TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOStreamOperator \(\)](#)
- void [TestGetters \(\)](#)
- [mtk::Real ScalarField \(const mtk::Real &xx, const mtk::Real &yy, const mtk::Real &zz\)](#)
- void [TestBindScalarFieldWriteToFile \(\)](#)
- int [main \(\)](#)

18.171.1 Detailed Description

Author

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

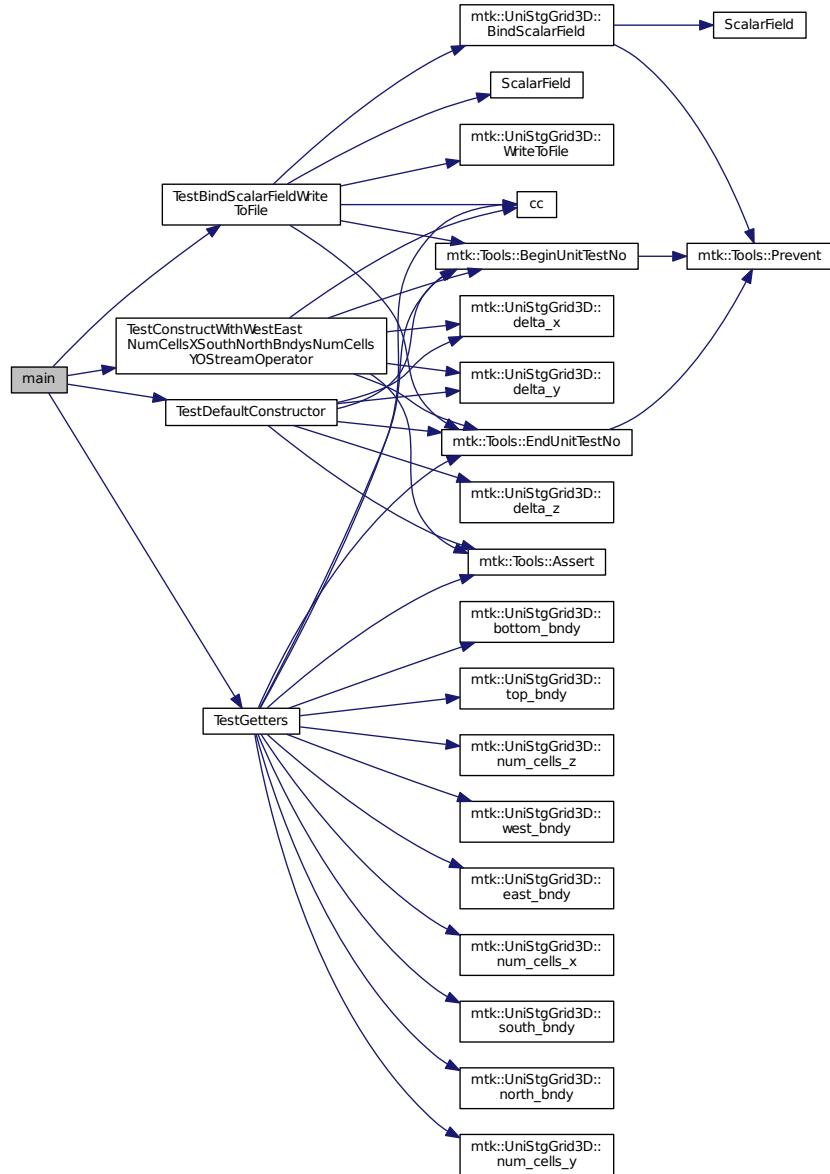
Definition in file [mtk_uni_stg_grid_3d_test.cc](#).

18.171.2 Function Documentation

18.171.2.1 int main ()

Definition at line 168 of file [mtk_uni_stg_grid_3d_test.cc](#).

Here is the call graph for this function:



18.171.2.2 `mtk::Real ScalarField (const mtk::Real & xx, const mtk::Real & yy, const mtk::Real & zz)`

Definition at line 131 of file [mtk_uni_stg_grid_3d_test.cc](#).

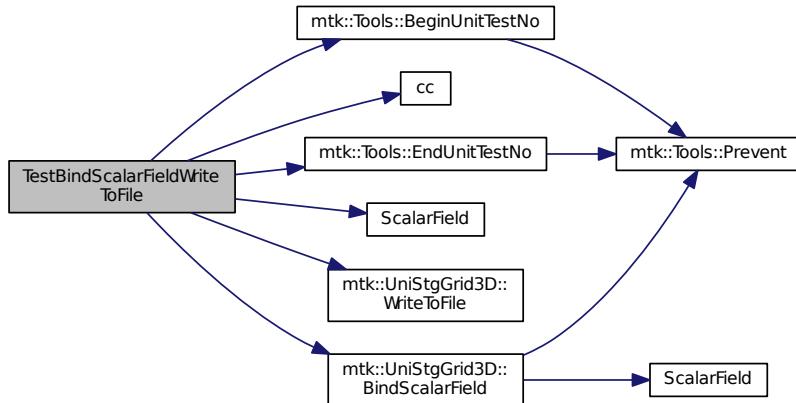
Here is the caller graph for this function:



18.171.2.3 void TestBindScalarFieldWriteToFile()

Definition at line 138 of file [mtk_uni_stg_grid_3d_test.cc](#).

Here is the call graph for this function:



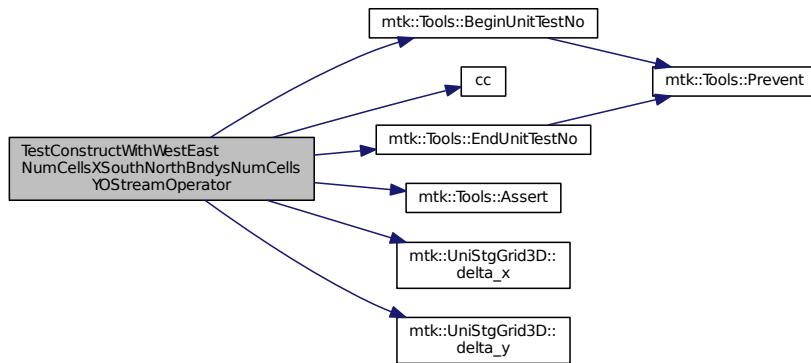
Here is the caller graph for this function:



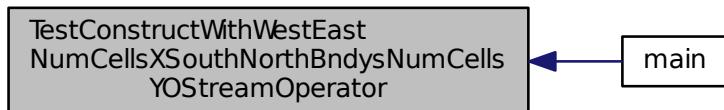
18.171.2.4 void TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOStreamOperator()

Definition at line 74 of file [mtk_uni_stg_grid_3d_test.cc](#).

Here is the call graph for this function:



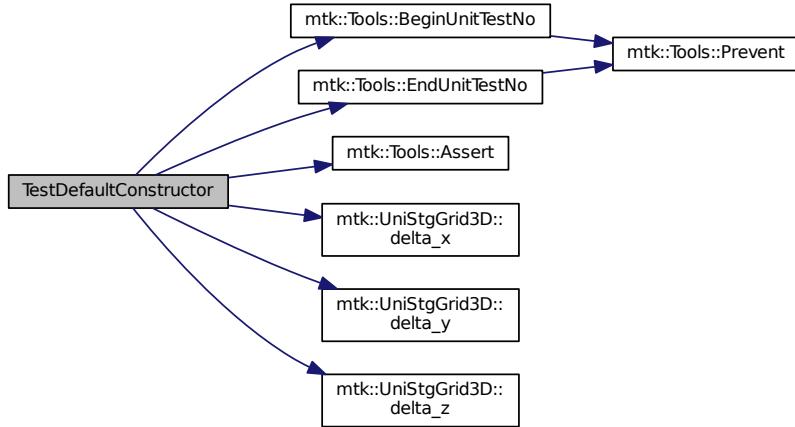
Here is the caller graph for this function:



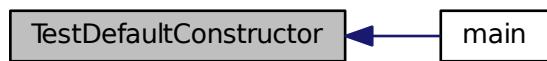
18.171.2.5 void TestDefaultConstructor()

Definition at line 61 of file [mtk_uni_stg_grid_3d_test.cc](#).

Here is the call graph for this function:



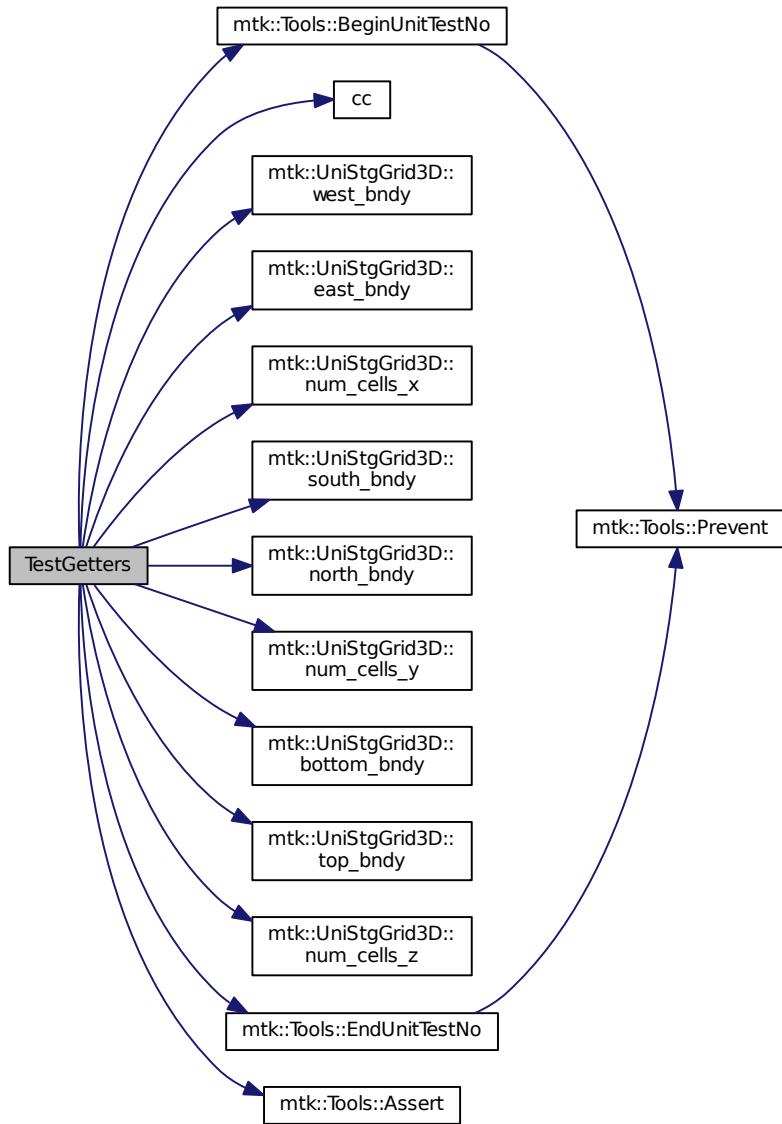
Here is the caller graph for this function:



18.171.2.6 void TestGetters()

Definition at line 98 of file [mtk_uni_stg_grid_3d_test.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



18.172 mtk_uni_stg_grid_3d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2016, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.cscc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.cscc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #include <cmath>
00055 #include <ctime>
00056
00057 #include <iostream>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062     mtk::Tools::BeginUnitTests(1);
00063
  
```

```

00064
00065     mtk::UniStgGrid3D gg;
00066
00067     mtk::Tools::EndUnitTestNo(1);
00068     mtk::Tools::Assert(gg.delta_x() == mtk::kZero &&
00069                         gg.delta_y() == mtk::kZero &&
00070                         gg.delta_z() == mtk::kZero);
00071 }
00072
00073 void
00074 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator
00075 () {
00076     mtk::Tools::BeginUnitTestNo(2);
00077
00078     mtk::Real aa = 0.0;
00079     mtk::Real bb = 1.0;
00080     mtk::Real cc = 0.0;
00081     mtk::Real dd = 1.0;
00082     mtk::Real ee = 0.0;
00083     mtk::Real ff = 1.0;
00084
00085     int nn = 5;
00086     int mm = 7;
00087     int oo = 7;
00088
00089     mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00090
00091     std::cout << gg << std::endl;
00092
00093     mtk::Tools::EndUnitTestNo(2);
00094     mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00095                         abs(gg.delta_y() - 0.142857) <
00096                         mtk::kDefaultTolerance);
00097 }
00098 void TestGetters() {
00099
00100    mtk::Tools::BeginUnitTestNo(3);
00101
00102    mtk::Real aa = 0.0;
00103    mtk::Real bb = 1.0;
00104    mtk::Real cc = 0.0;
00105    mtk::Real dd = 1.0;
00106    mtk::Real ee = 0.0;
00107    mtk::Real ff = 1.0;
00108
00109    int nn = 5;
00110    int mm = 7;
00111    int oo = 6;
00112
00113    mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00114
00115    bool assertion{true};
00116
00117    assertion = assertion && (gg.west_bndy() == aa);
00118    assertion = assertion && (gg.east_bndy() == bb);
00119    assertion = assertion && (gg.num_cells_x() == nn);
00120    assertion = assertion && (gg.south_bndy() == cc);
00121    assertion = assertion && (gg.north_bndy() == dd);
00122    assertion = assertion && (gg.num_cells_y() == mm);
00123    assertion = assertion && (gg.bottom_bndy() == ee);
00124    assertion = assertion && (gg.top_bndy() == ff);
00125    assertion = assertion && (gg.num_cells_z() == oo);
00126
00127    mtk::Tools::EndUnitTestNo(3);
00128    mtk::Tools::Assert(assertion);
00129 }
00130
00131 mtk::Real ScalarField(const mtk::Real &xx,
00132                         const mtk::Real &yy,
00133                         const mtk::Real &zz) {
00134
00135     return xx + yy + zz;
00136 }
00137
00138 void TestBindScalarFieldWriteToFile() {
00139
00140     mtk::Tools::BeginUnitTestNo(4);
00141
00142     mtk::Real aa = 0.0;

```

```
00143     mtk::Real bb = 1.0;
00144     mtk::Real cc = 0.0;
00145     mtk::Real dd = 1.0;
00146     mtk::Real ee = 0.0;
00147     mtk::Real ff = 1.0;
00148
00149     int nn = 50;
00150     int mm = 50;
00151     int oo = 50;
00152
00153     mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00154
00155     gg.BindScalarField(ScalarField);
00156
00157     if(!gg.WriteToFile("mtk_uni_stg_grid_3d_test_04_gg.dat",
00158                         "x",
00159                         "y",
00160                         "z",
00161                         "u(x,y,z)")) {
00162         std::cerr << "Error writing to file." << std::endl;
00163     }
00164
00165     mtk::Tools::EndUnitTestNo(4);
00166 }
00167
00168 int main () {
00169
00170     std::cout << "Testing mtk::UniStgGrid3D class." << std::endl;
00171
00172     TestDefaultConstructor();
00173     TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSstreamOperator
00174     ();
00175     TestGetters();
00176 }
```

Index

BANDED
 Enumerations., [37](#)

COL_MAJOR
 Enumerations., [37](#)

CRS
 Enumerations., [37](#)

CURL
 Enumerations., [36](#)

DENSE
 Enumerations., [37](#)

DIVERGENCE
 Enumerations., [36](#)

Data Structures., [39](#)

 Enumerations., [36](#)
 BANDED, [37](#)
 COL_MAJOR, [37](#)
 CRS, [37](#)
 CURL, [36](#)
 DENSE, [37](#)
 DIVERGENCE, [36](#)
 GRADIENT, [36](#)
 INTERPOLATION, [36](#)
 LAPLACIAN, [37](#)
 NOOP, [36](#)
 ROW_MAJOR, [37](#)
 SCALAR, [37](#)
 SCALAR_TO_VECTOR, [36](#)
 VECTOR, [37](#)
 VECTOR_TO_SCALAR, [36](#)

Execution Tools., [38](#)

 Foundations., [33](#)
 Real, [34](#)

GRADIENT
 Enumerations., [36](#)

Grids., [41](#)

INTERPOLATION
 Enumerations., [36](#)

LAPLACIAN
 Enumerations., [37](#)

Mimetic Operators., [42](#)

mtk, [45](#)

operator<<, [48, 49](#)

NOOP
 Enumerations., [36](#)

Numerical Methods., [40](#)

operator<<
 mtk, [48, 49](#)

ROW_MAJOR
 Enumerations., [37](#)

Real
 Foundations., [34](#)

SCALAR
 Enumerations., [37](#)

SCALAR_TO_VECTOR
 Enumerations., [36](#)

VECTOR
 Enumerations., [37](#)

VECTOR_TO_SCALAR
 Enumerations., [36](#)