

MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.8

Mon Feb 1 2016 17:07:18

Contents

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

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

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

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

| | |
|-------------------------------------------------------------|----|
| 17.4.1 Detailed Description | 89 |
| 17.4.2 Constructor & Destructor Documentation | 89 |
| 17.4.2.1 Div1D | 89 |
| 17.4.2.2 Div1D | 89 |
| 17.4.2.3 ~Div1D | 90 |
| 17.4.3 Member Function Documentation | 90 |
| 17.4.3.1 AssembleOperator | 90 |
| 17.4.3.2 coeffs_interior | 90 |
| 17.4.3.3 ComputePreliminaryApproximations | 90 |
| 17.4.3.4 ComputeRationalBasisNullSpace | 91 |
| 17.4.3.5 ComputeStencilBoundaryGrid | 92 |
| 17.4.3.6 ComputeStencilInteriorGrid | 92 |
| 17.4.3.7 ComputeWeights | 93 |
| 17.4.3.8 ConstructDiv1D | 94 |
| 17.4.3.9 mim_bndy | 94 |
| 17.4.3.10 num_bndy_coeffs | 95 |
| 17.4.3.11 ReturnAsDenseMatrix | 95 |
| 17.4.3.12 ReturnAsDimensionlessDenseMatrix | 96 |
| 17.4.3.13 sums_rows_mim_bndy | 96 |
| 17.4.3.14 weights_cbs | 96 |
| 17.4.3.15 weights_crs | 97 |
| 17.4.4 Friends And Related Function Documentation | 97 |
| 17.4.4.1 operator<< | 97 |
| 17.4.5 Member Data Documentation | 97 |
| 17.4.5.1 coeffs_interior_ | 97 |
| 17.4.5.2 dim_null_ | 97 |
| 17.4.5.3 divergence_ | 97 |
| 17.4.5.4 divergence_length_ | 97 |
| 17.4.5.5 mim_bndy_ | 97 |
| 17.4.5.6 mimetic_threshold_ | 97 |
| 17.4.5.7 minrow_ | 98 |
| 17.4.5.8 num_bndy_coeffs_ | 98 |
| 17.4.5.9 order_accuracy_ | 98 |
| 17.4.5.10 prem_apps_ | 98 |
| 17.4.5.11 rat_basis_null_space_ | 98 |
| 17.4.5.12 row_ | 98 |
| 17.4.5.13 sums_rows_mim_bndy_ | 98 |

| | | |
|-----------|--------------------------------------------------|-----|
| 17.4.5.14 | weights_cbs_ | 98 |
| 17.4.5.15 | weights_crs_ | 98 |
| 17.5 | mtk::Div2D Class Reference | 98 |
| 17.5.1 | Detailed Description | 100 |
| 17.5.2 | Constructor & Destructor Documentation | 100 |
| 17.5.2.1 | Div2D | 100 |
| 17.5.2.2 | Div2D | 100 |
| 17.5.2.3 | ~Div2D | 100 |
| 17.5.3 | Member Function Documentation | 101 |
| 17.5.3.1 | ConstructDiv2D | 101 |
| 17.5.3.2 | ReturnAsDenseMatrix | 101 |
| 17.5.4 | Member Data Documentation | 102 |
| 17.5.4.1 | divergence_ | 102 |
| 17.5.4.2 | mimetic_threshold_ | 102 |
| 17.5.4.3 | order_accuracy_ | 102 |
| 17.6 | mtk::Div3D Class Reference | 102 |
| 17.6.1 | Detailed Description | 104 |
| 17.6.2 | Constructor & Destructor Documentation | 104 |
| 17.6.2.1 | Div3D | 104 |
| 17.6.2.2 | Div3D | 104 |
| 17.6.2.3 | ~Div3D | 104 |
| 17.6.3 | Member Function Documentation | 105 |
| 17.6.3.1 | ConstructDiv3D | 105 |
| 17.6.3.2 | ReturnAsDenseMatrix | 106 |
| 17.6.4 | Member Data Documentation | 106 |
| 17.6.4.1 | divergence_ | 106 |
| 17.6.4.2 | mimetic_threshold_ | 106 |
| 17.6.4.3 | order_accuracy_ | 106 |
| 17.7 | mtk::GLPKAdapter Class Reference | 107 |
| 17.7.1 | Detailed Description | 107 |
| 17.7.2 | Member Function Documentation | 107 |
| 17.7.2.1 | SolveSimplexAndCompare | 108 |
| 17.8 | mtk::Grad1D Class Reference | 110 |
| 17.8.1 | Detailed Description | 113 |
| 17.8.2 | Constructor & Destructor Documentation | 114 |
| 17.8.2.1 | Grad1D | 114 |
| 17.8.2.2 | Grad1D | 114 |

| | | |
|-----------|------------------------------------------------------------|-----|
| 17.8.2.3 | ~Grad1D | 114 |
| 17.8.3 | Member Function Documentation | 114 |
| 17.8.3.1 | AssembleOperator | 114 |
| 17.8.3.2 | coeffs_interior | 114 |
| 17.8.3.3 | ComputePreliminaryApproximations | 114 |
| 17.8.3.4 | ComputeRationalBasisNullSpace | 115 |
| 17.8.3.5 | ComputeStencilBoundaryGrid | 116 |
| 17.8.3.6 | ComputeStencilInteriorGrid | 116 |
| 17.8.3.7 | ComputeWeights | 117 |
| 17.8.3.8 | ConstructGrad1D | 117 |
| 17.8.3.9 | mim_bndy | 118 |
| 17.8.3.10 | num_bndy_coeffs | 119 |
| 17.8.3.11 | ReturnAsDenseMatrix | 119 |
| 17.8.3.12 | ReturnAsDenseMatrix | 120 |
| 17.8.3.13 | ReturnAsDimensionlessDenseMatrix | 121 |
| 17.8.3.14 | sums_rows_mim_bndy | 121 |
| 17.8.3.15 | weights_cbs | 121 |
| 17.8.3.16 | weights_crs | 121 |
| 17.8.4 | Friends And Related Function Documentation | 121 |
| 17.8.4.1 | operator<< | 121 |
| 17.8.5 | Member Data Documentation | 122 |
| 17.8.5.1 | coeffs_interior_ | 122 |
| 17.8.5.2 | dim_null_ | 122 |
| 17.8.5.3 | gradient_ | 122 |
| 17.8.5.4 | gradient_length_ | 122 |
| 17.8.5.5 | mim_bndy_ | 122 |
| 17.8.5.6 | mimetic_threshold_ | 122 |
| 17.8.5.7 | minrow_ | 122 |
| 17.8.5.8 | num_bndy_approxs_ | 122 |
| 17.8.5.9 | num_bndy_coeffs_ | 122 |
| 17.8.5.10 | order_accuracy_ | 123 |
| 17.8.5.11 | prem_apps_ | 123 |
| 17.8.5.12 | rat_basis_null_space_ | 123 |
| 17.8.5.13 | row_ | 123 |
| 17.8.5.14 | sums_rows_mim_bndy_ | 123 |
| 17.8.5.15 | weights_cbs_ | 123 |
| 17.8.5.16 | weights_crs_ | 123 |

| | | |
|-----------|--------------------------------------------|-----|
| 17.9 | mtk::Grad2D Class Reference | 123 |
| 17.9.1 | Detailed Description | 125 |
| 17.9.2 | Constructor & Destructor Documentation | 125 |
| 17.9.2.1 | Grad2D | 125 |
| 17.9.2.2 | Grad2D | 125 |
| 17.9.2.3 | ~Grad2D | 125 |
| 17.9.3 | Member Function Documentation | 126 |
| 17.9.3.1 | ConstructGrad2D | 126 |
| 17.9.3.2 | ReturnAsDenseMatrix | 126 |
| 17.9.4 | Member Data Documentation | 127 |
| 17.9.4.1 | gradient_ | 127 |
| 17.9.4.2 | mimetic_threshold_ | 127 |
| 17.9.4.3 | order_accuracy_ | 127 |
| 17.10 | mtk::Grad3D Class Reference | 127 |
| 17.10.1 | Detailed Description | 129 |
| 17.10.2 | Constructor & Destructor Documentation | 129 |
| 17.10.2.1 | Grad3D | 129 |
| 17.10.2.2 | Grad3D | 129 |
| 17.10.2.3 | ~Grad3D | 129 |
| 17.10.3 | Member Function Documentation | 130 |
| 17.10.3.1 | ConstructGrad3D | 130 |
| 17.10.3.2 | ReturnAsDenseMatrix | 131 |
| 17.10.4 | Member Data Documentation | 131 |
| 17.10.4.1 | gradient_ | 131 |
| 17.10.4.2 | mimetic_threshold_ | 131 |
| 17.10.4.3 | order_accuracy_ | 131 |
| 17.11 | mtk::Interp1D Class Reference | 132 |
| 17.11.1 | Detailed Description | 133 |
| 17.11.2 | Constructor & Destructor Documentation | 133 |
| 17.11.2.1 | Interp1D | 133 |
| 17.11.2.2 | Interp1D | 133 |
| 17.11.2.3 | ~Interp1D | 133 |
| 17.11.3 | Member Function Documentation | 133 |
| 17.11.3.1 | coeffs_interior | 133 |
| 17.11.3.2 | ConstructInterp1D | 133 |
| 17.11.3.3 | ReturnAsDenseMatrix | 134 |
| 17.11.4 | Friends And Related Function Documentation | 134 |

| | |
|----------------------------------------------------|-----|
| 17.11.4.1 operator<< | 134 |
| 17.11.5 Member Data Documentation | 134 |
| 17.11.5.1 coeffs_interior_ | 134 |
| 17.11.5.2 dir_interp_ | 135 |
| 17.11.5.3 order_accuracy_ | 135 |
| 17.12mtk::Interp2D Class Reference | 135 |
| 17.12.1 Detailed Description | 137 |
| 17.12.2 Constructor & Destructor Documentation | 137 |
| 17.12.2.1 Interp2D | 137 |
| 17.12.2.2 Interp2D | 137 |
| 17.12.2.3 ~Interp2D | 137 |
| 17.12.3 Member Function Documentation | 137 |
| 17.12.3.1 ConstructInterp2D | 137 |
| 17.12.3.2 ReturnAsDenseMatrix | 138 |
| 17.12.4 Member Data Documentation | 138 |
| 17.12.4.1 interpolator_ | 138 |
| 17.12.4.2 mimetic_threshold_ | 138 |
| 17.12.4.3 order_accuracy_ | 138 |
| 17.13mtk::Lap1D Class Reference | 138 |
| 17.13.1 Detailed Description | 140 |
| 17.13.2 Constructor & Destructor Documentation | 141 |
| 17.13.2.1 Lap1D | 141 |
| 17.13.2.2 Lap1D | 141 |
| 17.13.2.3 ~Lap1D | 141 |
| 17.13.3 Member Function Documentation | 141 |
| 17.13.3.1 ConstructLap1D | 141 |
| 17.13.3.2 data | 142 |
| 17.13.3.3 delta | 143 |
| 17.13.3.4 mimetic_threshold | 143 |
| 17.13.3.5 order_accuracy | 143 |
| 17.13.3.6 ReturnAsDenseMatrix | 144 |
| 17.13.3.7 sums_rows_mim_bndy | 145 |
| 17.13.4 Friends And Related Function Documentation | 145 |
| 17.13.4.1 operator<< | 145 |
| 17.13.5 Member Data Documentation | 145 |
| 17.13.5.1 delta_ | 145 |
| 17.13.5.2 laplacian_ | 145 |

| | |
|----------------------------------------------------------|-----|
| 17.13.5.3 laplacian_length_ | 145 |
| 17.13.5.4 mimetic_threshold_ | 145 |
| 17.13.5.5 order_accuracy_ | 145 |
| 17.13.5.6 sums_rows_mim_bndy_ | 145 |
| 17.14mtk::Lap2D Class Reference | 146 |
| 17.14.1 Detailed Description | 147 |
| 17.14.2 Constructor & Destructor Documentation | 147 |
| 17.14.2.1 Lap2D | 147 |
| 17.14.2.2 Lap2D | 147 |
| 17.14.2.3 ~Lap2D | 147 |
| 17.14.3 Member Function Documentation | 148 |
| 17.14.3.1 ConstructLap2D | 148 |
| 17.14.3.2 data | 148 |
| 17.14.3.3 ReturnAsDenseMatrix | 149 |
| 17.14.4 Member Data Documentation | 149 |
| 17.14.4.1 laplacian_ | 149 |
| 17.14.4.2 mimetic_threshold_ | 149 |
| 17.14.4.3 order_accuracy_ | 149 |
| 17.15mtk::Lap3D Class Reference | 149 |
| 17.15.1 Detailed Description | 151 |
| 17.15.2 Constructor & Destructor Documentation | 151 |
| 17.15.2.1 Lap3D | 151 |
| 17.15.2.2 Lap3D | 151 |
| 17.15.2.3 ~Lap3D | 151 |
| 17.15.3 Member Function Documentation | 152 |
| 17.15.3.1 ConstructLap3D | 152 |
| 17.15.3.2 data | 152 |
| 17.15.3.3 operator* | 153 |
| 17.15.3.4 ReturnAsDenseMatrix | 153 |
| 17.15.4 Member Data Documentation | 153 |
| 17.15.4.1 laplacian_ | 153 |
| 17.15.4.2 mimetic_threshold_ | 153 |
| 17.15.4.3 order_accuracy_ | 153 |
| 17.16mtk::LAPACKAdapter Class Reference | 153 |
| 17.16.1 Detailed Description | 154 |
| 17.16.2 Member Function Documentation | 155 |
| 17.16.2.1 QRFactorDenseMatrix | 155 |

| | |
|----------------------------------------------------------|-----|
| 17.16.2.2 SolveDenseSystem | 156 |
| 17.16.2.3 SolveDenseSystem | 157 |
| 17.16.2.4 SolveDenseSystem | 158 |
| 17.16.2.5 SolveDenseSystem | 159 |
| 17.16.2.6 SolveRectangularDenseSystem | 160 |
| 17.17mtk::Matrix Class Reference | 161 |
| 17.17.1 Detailed Description | 164 |
| 17.17.2 Constructor & Destructor Documentation | 164 |
| 17.17.2.1 Matrix | 164 |
| 17.17.2.2 Matrix | 165 |
| 17.17.2.3 ~Matrix | 166 |
| 17.17.3 Member Function Documentation | 166 |
| 17.17.3.1 abs_density | 166 |
| 17.17.3.2 abs_sparsity | 166 |
| 17.17.3.3 bandwidth | 166 |
| 17.17.3.4 IncreaseNumNull | 166 |
| 17.17.3.5 IncreaseNumZero | 167 |
| 17.17.3.6 kl | 167 |
| 17.17.3.7 ku | 167 |
| 17.17.3.8 ld | 167 |
| 17.17.3.9 num_cols | 167 |
| 17.17.3.10num_non_null | 168 |
| 17.17.3.11num_non_zero | 168 |
| 17.17.3.12num_null | 168 |
| 17.17.3.13num_rows | 168 |
| 17.17.3.14num_values | 169 |
| 17.17.3.15num_zero | 169 |
| 17.17.3.16ordering | 169 |
| 17.17.3.17rel_density | 170 |
| 17.17.3.18rel_sparsity | 170 |
| 17.17.3.19set_num_cols | 170 |
| 17.17.3.20set_num_null | 171 |
| 17.17.3.21set_num_rows | 172 |
| 17.17.3.22set_num_zero | 172 |
| 17.17.3.23set_ordering | 173 |
| 17.17.3.24set_storage | 174 |
| 17.17.3.25storage | 174 |

| | |
|--------------------------------------------------------------|-----|
| 17.17.4 Member Data Documentation | 175 |
| 17.17.4.1 abs_density_ | 175 |
| 17.17.4.2 abs_sparsity_ | 175 |
| 17.17.4.3 bandwidth_ | 175 |
| 17.17.4.4 kl_ | 175 |
| 17.17.4.5 ku_ | 175 |
| 17.17.4.6 ld_ | 175 |
| 17.17.4.7 num_cols_ | 175 |
| 17.17.4.8 num_non_null_ | 176 |
| 17.17.4.9 num_non_zero_ | 176 |
| 17.17.4.10 num_null_ | 176 |
| 17.17.4.11 num_rows_ | 176 |
| 17.17.4.12 num_values_ | 176 |
| 17.17.4.13 num_zero_ | 176 |
| 17.17.4.14 ordering_ | 176 |
| 17.17.4.15 rel_density_ | 176 |
| 17.17.4.16 rel_sparsity_ | 176 |
| 17.17.4.17 storage_ | 176 |
| 17.18 mtk::Quad1D Class Reference | 177 |
| 17.18.1 Detailed Description | 178 |
| 17.18.2 Constructor & Destructor Documentation | 178 |
| 17.18.2.1 Quad1D | 178 |
| 17.18.2.2 Quad1D | 178 |
| 17.18.2.3 ~Quad1D | 179 |
| 17.18.3 Member Function Documentation | 179 |
| 17.18.3.1 degree_approximation | 179 |
| 17.18.3.2 Integrate | 179 |
| 17.18.3.3 weights | 179 |
| 17.18.4 Friends And Related Function Documentation | 179 |
| 17.18.4.1 operator<< | 179 |
| 17.18.5 Member Data Documentation | 179 |
| 17.18.5.1 degree_approximation_ | 179 |
| 17.18.5.2 weights_ | 179 |
| 17.19 mtk::RobinBCDescriptor1D Class Reference | 179 |
| 17.19.1 Detailed Description | 181 |
| 17.19.2 Constructor & Destructor Documentation | 182 |
| 17.19.2.1 RobinBCDescriptor1D | 182 |

| | |
|----------------------------------------------------------|-----|
| 17.19.2.2 RobinBCDescriptor1D | 182 |
| 17.19.2.3 ~RobinBCDescriptor1D | 182 |
| 17.19.3 Member Function Documentation | 182 |
| 17.19.3.1 highest_order_diff_east | 182 |
| 17.19.3.2 highest_order_diff_west | 182 |
| 17.19.3.3 ImposeOnGrid | 183 |
| 17.19.3.4 ImposeOnLaplacianMatrix | 183 |
| 17.19.3.5 PushBackEastCoeff | 184 |
| 17.19.3.6 PushBackWestCoeff | 185 |
| 17.19.3.7 set_east_condition | 185 |
| 17.19.3.8 set_west_condition | 186 |
| 17.19.4 Member Data Documentation | 186 |
| 17.19.4.1 east_coefficients_ | 186 |
| 17.19.4.2 east_condition_ | 186 |
| 17.19.4.3 highest_order_diff_east_ | 186 |
| 17.19.4.4 highest_order_diff_west_ | 186 |
| 17.19.4.5 west_coefficients_ | 187 |
| 17.19.4.6 west_condition_ | 187 |
| 17.20mtk::RobinBCDescriptor2D Class Reference | 187 |
| 17.20.1 Detailed Description | 191 |
| 17.20.2 Constructor & Destructor Documentation | 191 |
| 17.20.2.1 RobinBCDescriptor2D | 191 |
| 17.20.2.2 RobinBCDescriptor2D | 191 |
| 17.20.2.3 ~RobinBCDescriptor2D | 191 |
| 17.20.3 Member Function Documentation | 191 |
| 17.20.3.1 highest_order_diff_east | 191 |
| 17.20.3.2 highest_order_diff_north | 192 |
| 17.20.3.3 highest_order_diff_south | 192 |
| 17.20.3.4 highest_order_diff_west | 192 |
| 17.20.3.5 ImposeOnEastBoundaryNoSpace | 192 |
| 17.20.3.6 ImposeOnEastBoundaryWithSpace | 193 |
| 17.20.3.7 ImposeOnGrid | 194 |
| 17.20.3.8 ImposeOnLaplacianMatrix | 196 |
| 17.20.3.9 ImposeOnNorthBoundaryNoSpace | 196 |
| 17.20.3.10ImposeOnNorthBoundaryWithSpace | 197 |
| 17.20.3.11ImposeOnSouthBoundaryNoSpace | 198 |
| 17.20.3.12ImposeOnSouthBoundaryWithSpace | 199 |

| | | |
|------------|------------------------------------------|-----|
| 17.20.3.13 | ImposeOnWestBoundaryNoSpace | 200 |
| 17.20.3.14 | ImposeOnWestBoundaryWithSpace | 200 |
| 17.20.3.15 | PushBackEastCoeff | 201 |
| 17.20.3.16 | PushBackNorthCoeff | 201 |
| 17.20.3.17 | PushBackSouthCoeff | 202 |
| 17.20.3.18 | PushBackWestCoeff | 202 |
| 17.20.3.19 | set_east_condition | 203 |
| 17.20.3.20 | set_north_condition | 203 |
| 17.20.3.21 | set_south_condition | 204 |
| 17.20.3.22 | set_west_condition | 204 |
| 17.20.4 | Member Data Documentation | 205 |
| 17.20.4.1 | east_coefficients_ | 205 |
| 17.20.4.2 | east_condition_ | 205 |
| 17.20.4.3 | highest_order_diff_east_ | 205 |
| 17.20.4.4 | highest_order_diff_north_ | 205 |
| 17.20.4.5 | highest_order_diff_south_ | 205 |
| 17.20.4.6 | highest_order_diff_west_ | 205 |
| 17.20.4.7 | north_coefficients_ | 205 |
| 17.20.4.8 | north_condition_ | 205 |
| 17.20.4.9 | south_coefficients_ | 206 |
| 17.20.4.10 | south_condition_ | 206 |
| 17.20.4.11 | west_coefficients_ | 206 |
| 17.20.4.12 | west_condition_ | 206 |
| 17.21 | mtk::RobinBCDescriptor3D Class Reference | 206 |
| 17.21.1 | Detailed Description | 210 |
| 17.21.2 | Constructor & Destructor Documentation | 210 |
| 17.21.2.1 | RobinBCDescriptor3D | 210 |
| 17.21.2.2 | RobinBCDescriptor3D | 210 |
| 17.21.2.3 | ~RobinBCDescriptor3D | 210 |
| 17.21.3 | Member Function Documentation | 210 |
| 17.21.3.1 | highest_order_diff_west | 210 |
| 17.21.3.2 | ImposeOnEastBoundaryNoSpace | 210 |
| 17.21.3.3 | ImposeOnEastBoundaryWithSpace | 211 |
| 17.21.3.4 | ImposeOnGrid | 211 |
| 17.21.3.5 | ImposeOnLaplacianMatrix | 211 |
| 17.21.3.6 | ImposeOnNorthBoundaryNoSpace | 211 |
| 17.21.3.7 | ImposeOnNorthBoundaryWithSpace | 211 |

| | | |
|------------|-----------------------------------|-----|
| 17.21.3.8 | ImposeOnSouthBoundaryNoSpace | 212 |
| 17.21.3.9 | ImposeOnSouthBoundaryWithSpace | 212 |
| 17.21.3.10 | ImposeOnWestBoundaryNoSpace | 212 |
| 17.21.3.11 | ImposeOnWestBoundaryWithSpace | 212 |
| 17.21.3.12 | PushBackWestCoeff | 213 |
| 17.21.3.13 | set_west_condition | 213 |
| 17.21.4 | Member Data Documentation | 213 |
| 17.21.4.1 | bottom_coefficients_ | 213 |
| 17.21.4.2 | bottom_condition_ | 213 |
| 17.21.4.3 | east_coefficients_ | 213 |
| 17.21.4.4 | east_condition_ | 213 |
| 17.21.4.5 | highest_order_diff_bottom_ | 213 |
| 17.21.4.6 | highest_order_diff_east_ | 213 |
| 17.21.4.7 | highest_order_diff_north_ | 213 |
| 17.21.4.8 | highest_order_diff_south_ | 214 |
| 17.21.4.9 | highest_order_diff_top_ | 214 |
| 17.21.4.10 | highest_order_diff_west_ | 214 |
| 17.21.4.11 | north_coefficients_ | 214 |
| 17.21.4.12 | north_condition_ | 214 |
| 17.21.4.13 | south_coefficients_ | 214 |
| 17.21.4.14 | south_condition_ | 214 |
| 17.21.4.15 | top_coefficients_ | 214 |
| 17.21.4.16 | top_condition_ | 214 |
| 17.21.4.17 | west_coefficients_ | 214 |
| 17.21.4.18 | west_condition_ | 215 |
| 17.22 | mtk::Tools Class Reference | 215 |
| 17.22.1 | Detailed Description | 216 |
| 17.22.2 | Member Function Documentation | 216 |
| 17.22.2.1 | Assert | 216 |
| 17.22.2.2 | BeginUnitTestNo | 216 |
| 17.22.2.3 | EndUnitTestNo | 216 |
| 17.22.2.4 | Prevent | 217 |
| 17.22.3 | Member Data Documentation | 217 |
| 17.22.3.1 | begin_time_ | 217 |
| 17.22.3.2 | duration_ | 217 |
| 17.22.3.3 | test_number_ | 217 |
| 17.23 | mtk::UniStgGrid1D Class Reference | 218 |

| | |
|----------------------------------------------------|-----|
| 17.23.1 Detailed Description | 220 |
| 17.23.2 Constructor & Destructor Documentation | 220 |
| 17.23.2.1 UniStgGrid1D | 220 |
| 17.23.2.2 UniStgGrid1D | 220 |
| 17.23.2.3 UniStgGrid1D | 220 |
| 17.23.2.4 ~UniStgGrid1D | 221 |
| 17.23.3 Member Function Documentation | 221 |
| 17.23.3.1 BindScalarField | 221 |
| 17.23.3.2 BindVectorField | 221 |
| 17.23.3.3 delta_x | 222 |
| 17.23.3.4 discrete_domain_x | 222 |
| 17.23.3.5 discrete_field | 222 |
| 17.23.3.6 east_bndy_x | 223 |
| 17.23.3.7 num_cells_x | 223 |
| 17.23.3.8 west_bndy_x | 224 |
| 17.23.3.9 WriteToFile | 224 |
| 17.23.4 Friends And Related Function Documentation | 224 |
| 17.23.4.1 operator<< | 225 |
| 17.23.5 Member Data Documentation | 225 |
| 17.23.5.1 delta_x_ | 225 |
| 17.23.5.2 discrete_domain_x_ | 225 |
| 17.23.5.3 discrete_field_ | 225 |
| 17.23.5.4 east_bndy_x_ | 225 |
| 17.23.5.5 nature_ | 225 |
| 17.23.5.6 num_cells_x_ | 225 |
| 17.23.5.7 west_bndy_x_ | 225 |
| 17.24mtk::UniStgGrid2D Class Reference | 225 |
| 17.24.1 Detailed Description | 228 |
| 17.24.2 Constructor & Destructor Documentation | 229 |
| 17.24.2.1 UniStgGrid2D | 229 |
| 17.24.2.2 UniStgGrid2D | 229 |
| 17.24.2.3 UniStgGrid2D | 229 |
| 17.24.2.4 ~UniStgGrid2D | 229 |
| 17.24.3 Member Function Documentation | 230 |
| 17.24.3.1 BindScalarField | 230 |
| 17.24.3.2 BindVectorField | 230 |
| 17.24.3.3 BindVectorFieldPComponent | 231 |

| | |
|--------------------------------------------------------------|-----|
| 17.24.3.4 BindVectorFieldQComponent | 231 |
| 17.24.3.5 Bound | 231 |
| 17.24.3.6 delta_x | 232 |
| 17.24.3.7 delta_y | 232 |
| 17.24.3.8 discrete_domain_x | 233 |
| 17.24.3.9 discrete_domain_y | 233 |
| 17.24.3.10 discrete_field | 234 |
| 17.24.3.11 east_bndy | 234 |
| 17.24.3.12 nature | 235 |
| 17.24.3.13 north_bndy | 236 |
| 17.24.3.14 num_cells_x | 236 |
| 17.24.3.15 num_cells_y | 237 |
| 17.24.3.16 Size | 238 |
| 17.24.3.17 south_bndy | 238 |
| 17.24.3.18 west_bndy | 239 |
| 17.24.3.19 WriteToFile | 240 |
| 17.24.4 Friends And Related Function Documentation | 241 |
| 17.24.4.1 operator<< | 241 |
| 17.24.5 Member Data Documentation | 241 |
| 17.24.5.1 delta_x_ | 241 |
| 17.24.5.2 delta_y_ | 241 |
| 17.24.5.3 discrete_domain_x_ | 241 |
| 17.24.5.4 discrete_domain_y_ | 241 |
| 17.24.5.5 discrete_field_ | 242 |
| 17.24.5.6 east_bndy_ | 242 |
| 17.24.5.7 nature_ | 242 |
| 17.24.5.8 north_bndy_ | 242 |
| 17.24.5.9 num_cells_x_ | 242 |
| 17.24.5.10 num_cells_y_ | 242 |
| 17.24.5.11 south_bndy_ | 242 |
| 17.24.5.12 west_bndy_ | 242 |
| 17.25 mtk::UniStgGrid3D Class Reference | 242 |
| 17.25.1 Detailed Description | 246 |
| 17.25.2 Constructor & Destructor Documentation | 246 |
| 17.25.2.1 UniStgGrid3D | 246 |
| 17.25.2.2 UniStgGrid3D | 246 |
| 17.25.2.3 UniStgGrid3D | 246 |

| | |
|--------------------------------------------------------------|-----|
| 17.25.2.4 ~UniStgGrid3D | 247 |
| 17.25.3 Member Function Documentation | 247 |
| 17.25.3.1 BindScalarField | 247 |
| 17.25.3.2 BindVectorField | 248 |
| 17.25.3.3 BindVectorFieldPComponent | 248 |
| 17.25.3.4 BindVectorFieldQComponent | 249 |
| 17.25.3.5 BindVectorFieldRComponent | 249 |
| 17.25.3.6 bottom_bndy | 249 |
| 17.25.3.7 Bound | 250 |
| 17.25.3.8 delta_x | 250 |
| 17.25.3.9 delta_y | 250 |
| 17.25.3.10 delta_z | 250 |
| 17.25.3.11 discrete_domain_x | 250 |
| 17.25.3.12 discrete_domain_y | 250 |
| 17.25.3.13 discrete_domain_z | 251 |
| 17.25.3.14 discrete_field | 251 |
| 17.25.3.15 east_bndy | 251 |
| 17.25.3.16 nature | 251 |
| 17.25.3.17 north_bndy | 251 |
| 17.25.3.18 num_cells_x | 252 |
| 17.25.3.19 num_cells_y | 252 |
| 17.25.3.20 num_cells_z | 252 |
| 17.25.3.21 operator= | 253 |
| 17.25.3.22 Size | 253 |
| 17.25.3.23 south_bndy | 253 |
| 17.25.3.24 top_bndy | 253 |
| 17.25.3.25 west_bndy | 254 |
| 17.25.3.26 WriteToFile | 254 |
| 17.25.4 Friends And Related Function Documentation | 255 |
| 17.25.4.1 operator<< | 255 |
| 17.25.5 Member Data Documentation | 255 |
| 17.25.5.1 bottom_bndy_ | 255 |
| 17.25.5.2 delta_x_ | 255 |
| 17.25.5.3 delta_y_ | 255 |
| 17.25.5.4 delta_z_ | 255 |
| 17.25.5.5 discrete_domain_x_ | 255 |
| 17.25.5.6 discrete_domain_y_ | 255 |

| | |
|---------------------------------------------------------------------------------------------|------------|
| 17.25.5.7 discrete_domain_z_ | 255 |
| 17.25.5.8 discrete_field_ | 256 |
| 17.25.5.9 east_bndy_ | 256 |
| 17.25.5.10 nature_ | 256 |
| 17.25.5.11 north_bndy_ | 256 |
| 17.25.5.12 num_cells_x_ | 256 |
| 17.25.5.13 num_cells_y_ | 256 |
| 17.25.5.14 num_cells_z_ | 256 |
| 17.25.5.15 south_bndy_ | 256 |
| 17.25.5.16 top_bndy_ | 256 |
| 17.25.5.17 west_bndy_ | 256 |
| 18 File Documentation | 257 |
| 18.1 examples/curl_2d_angular_velocity/curl_2d_angular_velocity.cc File Reference | 257 |
| 18.1.1 Detailed Description | 257 |
| 18.1.2 Function Documentation | 258 |
| 18.1.2.1 main | 258 |
| 18.2 curl_2d_angular_velocity.cc | 258 |
| 18.3 examples/diffusion_3d/diffusion_3d.cc File Reference | 259 |
| 18.3.1 Detailed Description | 260 |
| 18.3.2 Function Documentation | 260 |
| 18.3.2.1 main | 260 |
| 18.4 diffusion_3d.cc | 260 |
| 18.5 examples/divergence_operators_1d/divergence_operators_1d.cc File Reference | 262 |
| 18.5.1 Detailed Description | 262 |
| 18.5.2 Function Documentation | 262 |
| 18.5.2.1 main | 262 |
| 18.6 divergence_operators_1d.cc | 262 |
| 18.7 examples/gradient_operators_1d/gradient_operators_1d.cc File Reference | 264 |
| 18.7.1 Detailed Description | 264 |
| 18.7.2 Function Documentation | 264 |
| 18.7.2.1 main | 264 |
| 18.8 gradient_operators_1d.cc | 265 |
| 18.9 examples/laplacian_operators_1d/laplacian_operators_1d.cc File Reference | 266 |
| 18.9.1 Detailed Description | 266 |
| 18.9.2 Function Documentation | 267 |
| 18.9.2.1 main | 267 |

| | |
|-------------------------------------------------------------------------------------------|-----|
| 18.10laplacian_operators_1d.cc | 267 |
| 18.11examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference | 268 |
| 18.11.1 Detailed Description | 269 |
| 18.11.2 Function Documentation | 269 |
| 18.11.2.1 main | 269 |
| 18.12minimalistic_poisson_1d.cc | 269 |
| 18.13examples/poisson_1d/poisson_1d.cc File Reference | 271 |
| 18.13.1 Detailed Description | 272 |
| 18.13.2 Function Documentation | 272 |
| 18.13.2.1 main | 272 |
| 18.14poisson_1d.cc | 272 |
| 18.15examples/poisson_2d/poisson_2d.cc File Reference | 275 |
| 18.15.1 Detailed Description | 276 |
| 18.15.2 Function Documentation | 276 |
| 18.15.2.1 main | 276 |
| 18.16poisson_2d.cc | 276 |
| 18.17examples/positive_weights_1d/positive_weights_1d.cc File Reference | 279 |
| 18.17.1 Detailed Description | 279 |
| 18.17.2 Function Documentation | 280 |
| 18.17.2.1 main | 280 |
| 18.18positive_weights_1d.cc | 280 |
| 18.19include/mtk.h File Reference | 281 |
| 18.19.1 Detailed Description | 282 |
| 18.20mtk.h | 283 |
| 18.21include/mtk_blas_adapter.h File Reference | 284 |
| 18.21.1 Detailed Description | 285 |
| 18.22mtk_blas_adapter.h | 285 |
| 18.23include/mtk_curl_2d.h File Reference | 286 |
| 18.23.1 Detailed Description | 288 |
| 18.24mtk_curl_2d.h | 288 |
| 18.25include/mtk_dense_matrix.h File Reference | 289 |
| 18.25.1 Detailed Description | 290 |
| 18.26mtk_dense_matrix.h | 290 |
| 18.27include/mtk_div_1d.h File Reference | 292 |
| 18.27.1 Detailed Description | 293 |
| 18.28mtk_div_1d.h | 293 |
| 18.29include/mtk_div_2d.h File Reference | 295 |

| | |
|--------------------------------------------------|-----|
| 18.29.1 Detailed Description | 296 |
| 18.30mtk_div_2d.h | 296 |
| 18.31include/mtk_div_3d.h File Reference | 297 |
| 18.31.1 Detailed Description | 299 |
| 18.32mtk_div_3d.h | 299 |
| 18.33include/mtk_enums.h File Reference | 300 |
| 18.33.1 Detailed Description | 301 |
| 18.34mtk_enums.h | 301 |
| 18.35include/mtk_glpk_adapter.h File Reference | 302 |
| 18.35.1 Detailed Description | 303 |
| 18.36mtk_glpk_adapter.h | 303 |
| 18.37include/mtk_grad_1d.h File Reference | 304 |
| 18.37.1 Detailed Description | 305 |
| 18.38mtk_grad_1d.h | 305 |
| 18.39include/mtk_grad_2d.h File Reference | 307 |
| 18.39.1 Detailed Description | 309 |
| 18.40mtk_grad_2d.h | 309 |
| 18.41include/mtk_grad_3d.h File Reference | 310 |
| 18.41.1 Detailed Description | 311 |
| 18.42mtk_grad_3d.h | 311 |
| 18.43include/mtk_interp_1d.h File Reference | 312 |
| 18.43.1 Detailed Description | 314 |
| 18.44mtk_interp_1d.h | 314 |
| 18.45include/mtk_interp_2d.h File Reference | 315 |
| 18.45.1 Detailed Description | 316 |
| 18.46mtk_interp_2d.h | 317 |
| 18.47include/mtk_lap_1d.h File Reference | 318 |
| 18.47.1 Detailed Description | 319 |
| 18.48mtk_lap_1d.h | 319 |
| 18.49include/mtk_lap_2d.h File Reference | 320 |
| 18.49.1 Detailed Description | 322 |
| 18.50mtk_lap_2d.h | 322 |
| 18.51include/mtk_lap_3d.h File Reference | 323 |
| 18.51.1 Detailed Description | 325 |
| 18.52mtk_lap_3d.h | 325 |
| 18.53include/mtk_lapack_adapter.h File Reference | 326 |
| 18.53.1 Detailed Description | 327 |

| | | |
|---------|---------------------------------------------------------------------|-----|
| 18.54 | mtk_lapack_adapter.h | 327 |
| 18.55 | include/mtk_matrix.h File Reference | 329 |
| 18.55.1 | Detailed Description | 329 |
| 18.56 | mtk_matrix.h | 330 |
| 18.57 | include/mtk_quad_1d.h File Reference | 331 |
| 18.57.1 | Detailed Description | 332 |
| 18.58 | mtk_quad_1d.h | 333 |
| 18.59 | include/mtk_robin_bc_descriptor_1d.h File Reference | 334 |
| 18.59.1 | Detailed Description | 335 |
| 18.60 | mtk_robin_bc_descriptor_1d.h | 336 |
| 18.61 | include/mtk_robin_bc_descriptor_2d.h File Reference | 337 |
| 18.61.1 | Detailed Description | 339 |
| 18.62 | mtk_robin_bc_descriptor_2d.h | 339 |
| 18.63 | include/mtk_robin_bc_descriptor_3d.h File Reference | 341 |
| 18.63.1 | Detailed Description | 343 |
| 18.64 | mtk_robin_bc_descriptor_3d.h | 343 |
| 18.65 | include/mtk_roots.h File Reference | 345 |
| 18.65.1 | Detailed Description | 346 |
| 18.66 | mtk_roots.h | 347 |
| 18.67 | include/mtk_tools.h File Reference | 348 |
| 18.67.1 | Detailed Description | 348 |
| 18.68 | mtk_tools.h | 349 |
| 18.69 | include/mtk_uni_stg_grid_1d.h File Reference | 350 |
| 18.69.1 | Detailed Description | 351 |
| 18.70 | mtk_uni_stg_grid_1d.h | 351 |
| 18.71 | include/mtk_uni_stg_grid_2d.h File Reference | 352 |
| 18.71.1 | Detailed Description | 353 |
| 18.72 | mtk_uni_stg_grid_2d.h | 354 |
| 18.73 | include/mtk_uni_stg_grid_3d.h File Reference | 355 |
| 18.73.1 | Detailed Description | 357 |
| 18.74 | mtk_uni_stg_grid_3d.h | 357 |
| 18.75 | Makefile.inc File Reference | 359 |
| 18.76 | Makefile.inc | 359 |
| 18.77 | README.md File Reference | 362 |
| 18.78 | README.md | 362 |
| 18.79 | src/mtk_blas_adapter.cc File Reference | 364 |
| 18.79.1 | Detailed Description | 365 |

| | | |
|----------|----------------------------------------|-----|
| 18.80 | mtk_blas_adapter.cc | 366 |
| 18.81 | src/mtk_curl_2d.cc File Reference | 370 |
| 18.81.1 | Detailed Description | 370 |
| 18.82 | mtk_curl_2d.cc | 371 |
| 18.83 | src/mtk_dense_matrix.cc File Reference | 373 |
| 18.84 | mtk_dense_matrix.cc | 373 |
| 18.85 | src/mtk_div_1d.cc File Reference | 380 |
| 18.85.1 | Detailed Description | 381 |
| 18.86 | mtk_div_1d.cc | 382 |
| 18.87 | src/mtk_div_2d.cc File Reference | 400 |
| 18.87.1 | Detailed Description | 401 |
| 18.88 | mtk_div_2d.cc | 401 |
| 18.89 | src/mtk_div_3d.cc File Reference | 403 |
| 18.89.1 | Detailed Description | 403 |
| 18.90 | mtk_div_3d.cc | 404 |
| 18.91 | src/mtk_glpk_adapter.cc File Reference | 406 |
| 18.91.1 | Detailed Description | 407 |
| 18.92 | mtk_glpk_adapter.cc | 407 |
| 18.93 | src/mtk_grad_1d.cc File Reference | 411 |
| 18.93.1 | Detailed Description | 412 |
| 18.94 | mtk_grad_1d.cc | 412 |
| 18.95 | src/mtk_grad_2d.cc File Reference | 432 |
| 18.95.1 | Detailed Description | 432 |
| 18.96 | mtk_grad_2d.cc | 432 |
| 18.97 | src/mtk_grad_3d.cc File Reference | 434 |
| 18.97.1 | Detailed Description | 435 |
| 18.98 | mtk_grad_3d.cc | 435 |
| 18.99 | src/mtk_interp_1d.cc File Reference | 437 |
| 18.99.1 | Detailed Description | 438 |
| 18.100 | mtk_interp_1d.cc | 438 |
| 18.101 | src/mtk_lap_1d.cc File Reference | 440 |
| 18.101.1 | Detailed Description | 441 |
| 18.102 | mtk_lap_1d.cc | 441 |
| 18.103 | src/mtk_lap_2d.cc File Reference | 446 |
| 18.103.1 | Detailed Description | 446 |
| 18.104 | mtk_lap_2d.cc | 447 |
| 18.105 | src/mtk_lap_3d.cc File Reference | 448 |

| | |
|----------------------------------------------------------|-----|
| 18.105. Detailed Description | 449 |
| 18.106. mtk_lap_3d.cc | 449 |
| 18.107. src/mtk_lapack_adapter.cc File Reference | 450 |
| 18.107. Detailed Description | 451 |
| 18.108. mtk_lapack_adapter.cc | 452 |
| 18.109. src/mtk_matrix.cc File Reference | 459 |
| 18.109. Detailed Description | 460 |
| 18.110. mtk_matrix.cc | 460 |
| 18.111. src/mtk_robin_bc_descriptor_1d.cc File Reference | 464 |
| 18.111. Detailed Description | 464 |
| 18.112. mtk_robin_bc_descriptor_1d.cc | 465 |
| 18.113. src/mtk_robin_bc_descriptor_2d.cc File Reference | 467 |
| 18.113. Detailed Description | 468 |
| 18.114. mtk_robin_bc_descriptor_2d.cc | 469 |
| 18.115. src/mtk_tools.cc File Reference | 478 |
| 18.115. Detailed Description | 478 |
| 18.116. mtk_tools.cc | 478 |
| 18.117. src/mtk_uni_stg_grid_1d.cc File Reference | 480 |
| 18.117. Detailed Description | 480 |
| 18.118. mtk_uni_stg_grid_1d.cc | 481 |
| 18.119. src/mtk_uni_stg_grid_2d.cc File Reference | 484 |
| 18.119. Detailed Description | 485 |
| 18.120. mtk_uni_stg_grid_2d.cc | 485 |
| 18.121. src/mtk_uni_stg_grid_3d.cc File Reference | 491 |
| 18.121. Detailed Description | 492 |
| 18.122. mtk_uni_stg_grid_3d.cc | 492 |
| 18.123. tests/mtk_blas_adapter_test.cc File Reference | 498 |
| 18.123. Detailed Description | 498 |
| 18.123.2. Function Documentation | 498 |
| 18.123.2.1. main | 499 |
| 18.124. mtk_blas_adapter_test.cc | 499 |
| 18.125. tests/mtk_dense_matrix_test.cc File Reference | 500 |
| 18.125. Detailed Description | 501 |
| 18.125.2. Function Documentation | 501 |
| 18.125.2.1. main | 501 |
| 18.126. mtk_dense_matrix_test.cc | 501 |
| 18.127. tests/mtk_div_1d_test.cc File Reference | 505 |

| | |
|-------------------------------------------------------|-----|
| 18.127. Detailed Description | 505 |
| 18.127. Function Documentation | 506 |
| 18.127.2.1main | 506 |
| 18.128. mtk_div_1d_test.cc | 506 |
| 18.129. tests/mtk_div_2d_test.cc File Reference | 509 |
| 18.129. Detailed Description | 510 |
| 18.129. Function Documentation | 510 |
| 18.129.2.1main | 510 |
| 18.130. mtk_div_2d_test.cc | 510 |
| 18.131. tests/mtk_div_3d_test.cc File Reference | 512 |
| 18.131. Detailed Description | 513 |
| 18.131. Function Documentation | 513 |
| 18.131.2.1main | 513 |
| 18.132. mtk_div_3d_test.cc | 513 |
| 18.133. tests/mtk_glpk_adapter_test.cc File Reference | 515 |
| 18.133. Detailed Description | 515 |
| 18.133. Function Documentation | 515 |
| 18.133.2.1main | 515 |
| 18.134. mtk_glpk_adapter_test.cc | 516 |
| 18.135. tests/mtk_grad_1d_test.cc File Reference | 517 |
| 18.135. Detailed Description | 517 |
| 18.135. Function Documentation | 517 |
| 18.135.2.1main | 517 |
| 18.136. mtk_grad_1d_test.cc | 517 |
| 18.137. tests/mtk_grad_2d_test.cc File Reference | 521 |
| 18.137. Detailed Description | 522 |
| 18.137. Function Documentation | 522 |
| 18.137.2.1main | 522 |
| 18.138. mtk_grad_2d_test.cc | 522 |
| 18.139. tests/mtk_grad_3d_test.cc File Reference | 524 |
| 18.139. Detailed Description | 525 |
| 18.139. Function Documentation | 525 |
| 18.139.2.1main | 525 |
| 18.140. mtk_grad_3d_test.cc | 525 |
| 18.141. tests/mtk_interp_1d_test.cc File Reference | 527 |
| 18.141. Detailed Description | 527 |
| 18.141. Function Documentation | 527 |

| | |
|---------------------------------------------------------------|-----|
| 18.141.2.1main | 528 |
| 18.142mtk_interp_1d_test.cc | 528 |
| 18.143tests/mtk_lap_1d_test.cc File Reference | 529 |
| 18.143.1Detailed Description | 530 |
| 18.143.2Function Documentation | 530 |
| 18.143.2.1main | 530 |
| 18.144mtk_lap_1d_test.cc | 530 |
| 18.145tests/mtk_lap_2d_test.cc File Reference | 532 |
| 18.145.1Detailed Description | 533 |
| 18.145.2Function Documentation | 533 |
| 18.145.2.1main | 533 |
| 18.146mtk_lap_2d_test.cc | 533 |
| 18.147tests/mtk_lap_3d_test.cc File Reference | 535 |
| 18.147.1Detailed Description | 536 |
| 18.147.2Function Documentation | 536 |
| 18.147.2.1main | 536 |
| 18.148mtk_lap_3d_test.cc | 536 |
| 18.149tests/mtk_lapack_adapter_test.cc File Reference | 538 |
| 18.149.1Detailed Description | 538 |
| 18.149.2Function Documentation | 538 |
| 18.149.2.1main | 538 |
| 18.150mtk_lapack_adapter_test.cc | 539 |
| 18.151tests/mtk_robin_bc_descriptor_2d_test.cc File Reference | 540 |
| 18.151.1Detailed Description | 540 |
| 18.151.2Function Documentation | 540 |
| 18.151.2.1main | 540 |
| 18.152mtk_robin_bc_descriptor_2d_test.cc | 540 |
| 18.153tests/mtk_uni_stg_grid_1d_test.cc File Reference | 543 |
| 18.153.1Detailed Description | 543 |
| 18.153.2Function Documentation | 544 |
| 18.153.2.1main | 544 |
| 18.154mtk_uni_stg_grid_1d_test.cc | 544 |
| 18.155tests/mtk_uni_stg_grid_2d_test.cc File Reference | 546 |
| 18.155.1Detailed Description | 546 |
| 18.155.2Function Documentation | 547 |
| 18.155.2.1main | 547 |
| 18.156mtk_uni_stg_grid_2d_test.cc | 547 |

| | |
|---------------------------------------------------------------------|------------|
| 18.157.1 tests/mtk_uni_stg_grid_3d_test.cc File Reference | 549 |
| 18.157.2 Detailed Description | 550 |
| 18.157.3 Function Documentation | 550 |
| 18.157.2.1 main | 550 |
| 18.157.4 mtk_uni_stg_grid_3d_test.cc | 550 |
| Index | 554 |

Chapter 1

Introduction

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

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

1.1 MTK Concerns

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

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

In order of dependence these are:

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

1.2 MTK Wrappers

The MTK collection of wrappers is:

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

Others are being strongly considered.

1.3 Contact, Support and Credits

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

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

Currently the developers are:

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

1.3.1 Acknowledgements and Contributions

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

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

Chapter 2

Referencing This Work

Please reference this work as follows:

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

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


Chapter 3

Read Me File and Installation Instructions

The Mimetic Methods Toolkit (MTK)

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

1. Description

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

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

2. Dependencies

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

```
```  
$(HOME)/Libraries/
```
```

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

For Linux:

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

For OS X:

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

3. Installation

PART 1. CONFIGURATION OF THE MAKEFILE.

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

```
'''
$ make help
-----
Makefile for the MTK.

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

- testall: runs all the tests.

- gendoc: generates the documentation for the library.

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

PART 2. BUILD THE LIBRARY.

```
'''
$ make
'''

If successful you'll read (before building the 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](mailto:esanchez@mail.sdsu.edu)** - @ejspeiro
- Jose E. Castillo, PhD - [jcastillo at mail dot sdsu dot edu](mailto:jcastillo@mail.sdsu.edu)
- Guillermo F. Miranda, PhD - [unigrav at hotmail dot com](mailto:unigrav@hotmail.com)
- Christopher P. Paolini, PhD - [paolini at engineering dot sdsu dot edu](mailto:paolini@engineering.sdsu.edu)
- Angel Boada.
- Johnny Corbino.
- Raul Vargas-Navarro.

4.1. Acknowledgements and Contributions

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

- Mohammad Abouali, PhD
- Dany De Cecchis, PhD

- Otilio Rojas, PhD
- Julia Rossi.

5. Referencing This Work

Please reference this work as follows:

```

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

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

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

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

Thanks and happy coding!

Chapter 4

Programming Tools

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

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

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

Chapter 5

Tests and Test Architectures

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

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

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

Further architectures will be tested!

Chapter 6

User Manual, References and Theory

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

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

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

Chapter 7

Examples

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

Chapter 8

Licensing and Modifications

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

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

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

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

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

Chapter 9

Todo List

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

Implement Kronecker product using the BLAS.

Implement Kron using the BLAS.

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

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

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

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

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

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

Class `mtk::GLPKAdapter`

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

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

Review the definition of sparse matrices properties.

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

Review the definition of sparse matrices properties.

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

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

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

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

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

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

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

Check if this is the best way of stalling execution.

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

Review const-correctness of the pointer we return.

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

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

Member [mtk::UniStgGrid2D::discrete_domain_x \(\)](#) const

Review const-correctness of the pointer we return.

Member [mtk::UniStgGrid2D::discrete_domain_y \(\)](#) const

Review const-correctness of the pointer we return.

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

Review const-correctness of the pointer we return.

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

Review const-correctness of the pointer we return.

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

Review const-correctness of the pointer we return.

File [mtk_blas_adapter.cc](#)

Write documentation using LaTeX.

File [mtk_div_1d.cc](#)

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

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

File [mtk_glpk_adapter_test.cc](#)

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

File [mtk_grad_1d.cc](#)

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

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

File [mtk_lapack_adapter.cc](#)

Write documentation using LaTeX.

File [mtk_lapack_adapter_test.cc](#)

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

File [mtk_quad_1d.h](#)

Implement this class.

File [mtk_roots.h](#)

Test selective precision mechanisms.

File [mtk_uni_stg_grid_1d.h](#)

Create overloaded binding routines that read data from files.

File [mtk_uni_stg_grid_2d.h](#)

Create overloaded binding routines that read data from files.

File [mtk_uni_stg_grid_3d.h](#)

Create overloaded binding routines that read data from files.

Chapter 10

Bug List

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

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

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

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

Chapter 11

Module Index

11.1 Modules

Here is a list of all modules:

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

Chapter 12

Namespace Index

12.1 Namespace List

Here is a list of all namespaces with brief descriptions:

| | | |
|---------------------|---------------------------------------------|--------------------|
| 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 | 63 |
| mtk::DenseMatrix | Defines a common dense matrix, using a 1D array | 67 |
| mtk::Div1D | Implements a 1D mimetic divergence operator | 86 |
| mtk::Div2D | Implements a 2D mimetic divergence operator | 98 |
| mtk::Div3D | Implements a 3D mimetic divergence operator | 102 |
| mtk::GLPKAdapter | Adapter class for the GLPK API | 107 |
| mtk::Grad1D | Implements a 1D mimetic gradient operator | 110 |
| mtk::Grad2D | Implements a 2D mimetic gradient operator | 123 |
| mtk::Grad3D | Implements a 3D mimetic gradient operator | 127 |
| mtk::Interp1D | Implements a 1D interpolation operator | 132 |
| mtk::Interp2D | Implements a 2D interpolation operator | 135 |
| mtk::Lap1D | Implements a 1D mimetic Laplacian operator | 138 |
| mtk::Lap2D | Implements a 2D mimetic Laplacian operator | 146 |
| mtk::Lap3D | Implements a 3D mimetic Laplacian operator | 149 |
| mtk::LAPACKAdapter | Adapter class for the LAPACK API | 153 |
| mtk::Matrix | Definition of the representation of a matrix in the MTK | 161 |

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

Chapter 14

File Index

14.1 File List

Here is a list of all files with brief descriptions:

| | |
|--------------------------------------------------------------------------------|-----|
| Makefile.inc | 359 |
| examples/curl_2d/angular_velocity/ curl_2d_angular_velocity.cc | |
| Compute the curl of a 2D angular velocity field | 257 |
| examples/diffusion_3d/ diffusion_3d.cc | |
| Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs | 259 |
| examples/divergence_operators_1d/ divergence_operators_1d.cc | |
| Creates instances of a 1D divergence as computed by the CBS algorithm | 262 |
| examples/gradient_operators_1d/ gradient_operators_1d.cc | |
| Creates instances of a 1D gradient as computed by the CBS algorithm | 264 |
| examples/laplacian_operators_1d/ laplacian_operators_1d.cc | |
| Creates instances of a 1D Laplacian as computed by the CBS algorithm | 266 |
| examples/minimalistic_poisson_1d/ minimalistic_poisson_1d.cc | |
| Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs | 268 |
| examples/poisson_1d/ poisson_1d.cc | |
| Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs | 271 |
| examples/poisson_2d/ poisson_2d.cc | |
| Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs | 275 |
| examples/positive_weights_1d/ positive_weights_1d.cc | |
| The CBS algorithm computes positive-definite weights, for 1D operators | 279 |
| include/ mtk.h | |
| Includes the entire API | 281 |
| include/ mtk_blas_adapter.h | |
| Adapter class for the BLAS API | 284 |
| include/ mtk_curl_2d.h | |
| Includes the definition of the class Curl2D | 286 |
| include/ mtk_dense_matrix.h | |
| Defines a common dense matrix, using a 1D array | 289 |
| include/ mtk_div_1d.h | |
| Includes the definition of the class Div1D | 292 |
| include/ mtk_div_2d.h | |
| Includes the definition of the class Div2D | 295 |
| include/ mtk_div_3d.h | |
| Includes the definition of the class Div3D | 297 |

| | | |
|-------------------------------------------------------|------------------------------------------------------------------------------|-----|
| include/ mtk_enums.h | Considered enumeration types in the MTK | 300 |
| include/ mtk_glpk_adapter.h | Adapter class for the GLPK API | 302 |
| include/ mtk_grad_1d.h | Includes the definition of the class Grad1D | 304 |
| include/ mtk_grad_2d.h | Includes the definition of the class Grad2D | 307 |
| include/ mtk_grad_3d.h | Includes the definition of the class Grad3D | 310 |
| include/ mtk_interp_1d.h | Includes the definition of the class Interp1D | 312 |
| include/ mtk_interp_2d.h | Includes the definition of the class Interp2D | 315 |
| include/ mtk_lap_1d.h | Includes the definition of the class Lap1D | 318 |
| include/ mtk_lap_2d.h | Includes the implementation of the class Lap2D | 320 |
| include/ mtk_lap_3d.h | Includes the implementation of the class Lap3D | 323 |
| include/ mtk_lapack_adapter.h | Adapter class for the LAPACK API | 326 |
| include/ mtk_matrix.h | Definition of the representation of a matrix in the MTK | 329 |
| include/ mtk_quad_1d.h | Includes the definition of the class Quad1D | 331 |
| include/ mtk_robin_bc_descriptor_1d.h | Impose Robin boundary conditions on the operators and on the grids | 334 |
| include/ mtk_robin_bc_descriptor_2d.h | Impose Robin boundary conditions on the operators and on the grids | 337 |
| include/ mtk_robin_bc_descriptor_3d.h | Impose Robin boundary conditions on the operators and on the grids | 341 |
| include/ mtk_roots.h | Fundamental definitions to be used across all classes of the MTK | 345 |
| include/ mtk_tools.h | Tool manager class | 348 |
| include/ mtk_uni_stg_grid_1d.h | Definition of an 1D uniform staggered grid | 350 |
| include/ mtk_uni_stg_grid_2d.h | Definition of an 2D uniform staggered grid | 352 |
| include/ mtk_uni_stg_grid_3d.h | Definition of an 3D uniform staggered grid | 355 |
| src/ mtk_blas_adapter.cc | Adapter class for the BLAS API | 364 |
| src/ mtk_curl_2d.cc | Implements the class Curl2D | 370 |
| src/ mtk_dense_matrix.cc | | 373 |
| src/ mtk_div_1d.cc | Implements the class Div1D | 380 |
| src/ mtk_div_2d.cc | Implements the class Div2D | 400 |
| src/ mtk_div_3d.cc | Implements the class Div3D | 403 |

| | | |
|---------------------------------------------------|---------------------------------------------------------------------------------|-----|
| src/mtk_glpk_adapter.cc | Adapter class for the GLPK API | 406 |
| src/mtk_grad_1d.cc | Implements the class Grad1D | 411 |
| src/mtk_grad_2d.cc | Implements the class Grad2D | 432 |
| src/mtk_grad_3d.cc | Implements the class Grad3D | 434 |
| src/mtk_interp_1d.cc | Includes the implementation of the class Interp1D | 437 |
| src/mtk_lap_1d.cc | Includes the implementation of the class Lap1D | 440 |
| src/mtk_lap_2d.cc | Includes the implementation of the class Lap2D | 446 |
| src/mtk_lap_3d.cc | Includes the implementation of the class Lap3D | 448 |
| src/mtk_lapack_adapter.cc | Adapter class for the LAPACK API | 450 |
| src/mtk_matrix.cc | Implementing the representation of a matrix in the MTK | 459 |
| src/mtk_robin_bc_descriptor_1d.cc | Impose Robin boundary conditions on the operators and on the grids | 464 |
| src/mtk_robin_bc_descriptor_2d.cc | Impose Robin boundary conditions on the operators and on the grids | 467 |
| src/mtk_tools.cc | Tool manager class | 478 |
| src/mtk_uni_stg_grid_1d.cc | Implementation of an 1D uniform staggered grid | 480 |
| src/mtk_uni_stg_grid_2d.cc | Implementation of a 2D uniform staggered grid | 484 |
| src/mtk_uni_stg_grid_3d.cc | Implementation of a 3D uniform staggered grid | 491 |
| tests/mtk_blas_adapter_test.cc | Test file for the mtk::BLASAdapter class | 498 |
| tests/mtk_dense_matrix_test.cc | Test file for the mtk::DenseMatrix class | 500 |
| tests/mtk_div_1d_test.cc | Testing the mimetic 1D divergence, constructed with the CBS algorithm | 505 |
| tests/mtk_div_2d_test.cc | Test file for the mtk::Div2D class | 509 |
| tests/mtk_div_3d_test.cc | Test file for the mtk::Div3D class | 512 |
| tests/mtk_glpk_adapter_test.cc | Test file for the mtk::GLPKAdapter class | 515 |
| tests/mtk_grad_1d_test.cc | Testing the mimetic 1D gradient, constructed with the CBS algorithm | 517 |
| tests/mtk_grad_2d_test.cc | Test file for the mtk::Grad2D class | 521 |
| tests/mtk_grad_3d_test.cc | Test file for the mtk::Grad3D class | 524 |
| tests/mtk_interp_1d_test.cc | Testing the 1D interpolation | 527 |
| tests/mtk_lap_1d_test.cc | Testing the 1D Laplacian operator | 529 |

| | |
|----------------------------------------------------------------------------|-----|
| tests/ mtk_lap_2d_test.cc | |
| Test file for the mtk::Lap2D class | 532 |
| tests/ mtk_lap_3d_test.cc | |
| Test file for the mtk::Lap3D class | 535 |
| tests/ mtk_lapack_adapter_test.cc | |
| Test file for the mtk::LAPACKAdapter class | 538 |
| tests/ mtk_robin_bc_descriptor_2d_test.cc | |
| Test file for the mtk::RobinBCDescriptor2D class | 540 |
| tests/ mtk_uni_stg_grid_1d_test.cc | |
| Test file for the mtk::UniStgGrid1D class | 543 |
| tests/ mtk_uni_stg_grid_2d_test.cc | |
| Test file for the mtk::UniStgGrid2D class | 546 |
| tests/ mtk_uni_stg_grid_3d_test.cc | |
| Test file for the mtk::UniStgGrid3D class | 549 |

Chapter 15

Module Documentation

15.1 Roots.

Fundamental execution parameters and defined types.

Typedefs

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

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

Variables

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

15.1.1 Detailed Description

Fundamental execution parameters and defined types.

15.1.2 Typedef Documentation

15.1.2.1 `mtk::Real`

Warning

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

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

15.1.3 Variable Documentation

15.1.3.1 `mtk::kCriticalOrderAccuracyDiv {8}`

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

15.1.3.2 `mtk::kCriticalOrderAccuracyGrad {10}`

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

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

Warning

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

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

15.1.3.4 `mtk::kDefaultOrderAccuracy {2}`

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

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

Warning

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

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

15.1.3.6 `mtk::kOne {1.0f}`

Warning

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

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

15.1.3.7 mtk::kTwo {2.0f}

Warning

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

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

15.1.3.8 mtk::kZero {0.0f}

Warning

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

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

15.2 Enumerations.

Enumerations.

Enumerations

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

15.2.1 Detailed Description

Enumerations.

15.2.2 Enumeration Type Documentation

15.2.2.1 enum `mtk::DirInterp` [strong]

Used to tag different directions of interpolation supported.

Enumerator

SCALAR_TO_VECTOR Interpolations places scalar on vectors' location.

VECTOR_TO_SCALAR Interpolations places vectors on scalars' location.

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

15.2.2.2 enum `mtk::FieldNature` [strong]

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

See also

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

Enumerator

SCALAR Scalar-valued field.

VECTOR Vector-valued field.

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

15.2.2.3 `enum mtk::MatrixOrdering` [strong]

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

See also

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

Enumerator

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

COL_MAJOR Column-major ordering (Fortran).

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

15.2.2.4 `enum mtk::MatrixStorage` [strong]

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

Enumerator

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

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

CRS Compressed-Rows Storage: Must be implemented.

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

15.3 Execution tools.

Tools to ensure execution correctness.

Classes

- class [mtk::Tools](#)
Tool manager class.

15.3.1 Detailed Description

Tools to ensure execution correctness.

15.4 Data structures.

Fundamental data structures.

Classes

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

15.4.1 Detailed Description

Fundamental data structures.

15.5 Numerical methods.

Adapter classes and auxiliary numerical methods.

Classes

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

15.5.1 Detailed Description

Adapter classes and auxiliary numerical methods.

15.6 Grids.

Uniform rectangular staggered grids.

Classes

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

15.6.1 Detailed Description

Uniform rectangular staggered grids.

15.7 Mimetic operators.

Mimetic operators.

Classes

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

Typedefs

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

15.7.1 Detailed Description

Mimetic operators.

15.7.2 Typedef Documentation

15.7.2.1 `mtk::CoefficientFunction0D`

Warning

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

Definition at line 111 of file [mtk_robin_bc_descriptor_1d.h](#).

15.7.2.2 `mtk::CoefficientFunction1D`

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

15.7.2.3 `mtk::CoefficientFunction2D`

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

Chapter 16

Namespace Documentation

16.1 mtk Namespace Reference

Mimetic Methods Toolkit namespace.

Classes

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

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

Typedefs

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

Enumerations

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

Functions

- float [snrm2_](#) (int *n, float *x, int *incx)
- void [saxpy_](#) (int *n, float *sa, float *sx, int *incx, float *sy, int *incy)
- void [sgemv_](#) (char *trans, int *m, int *n, float *alpha, float *a, int *lda, float *x, int *incx, float *beta, float *y, int *incy)
- void [sgemm_](#) (char *transa, char *transb, int *m, int *n, int *k, double *alpha, double *a, int *lda, double *b, aamm int *ldb, double *beta, double *c, int *ldc)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::DenseMatrix](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Div1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Grad1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Interp1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::Lap1D](#) &in)
- void [sgesv_](#) (int *n, int *nrhs, [Real](#) *a, int *lda, int *ipiv, [Real](#) *b, int *ldb, int *info)
- void [sgels_](#) (char *trans, int *m, int *n, int *nrhs, [Real](#) *a, int *lda, [Real](#) *b, int *ldb, [Real](#) *work, int *lwork, int *info)
Single-precision GEneral matrix Least Squares solver.
- void [sgeqrf_](#) (int *m, int *n, [Real](#) *a, int *lda, [Real](#) *tau, [Real](#) *work, int *lwork, int *info)
Single-precision GEneral matrix QR Factorization.
- void [sormqr_](#) (char *side, char *trans, int *m, int *n, int *k, [Real](#) *a, int *lda, [Real](#) *tau, [Real](#) *c, int *ldc, [Real](#) *work, int *lwork, int *info)
Single-precision Orthogonal [Matrix](#) from QR factorization.
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid1D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid2D](#) &in)
- std::ostream & [operator<<](#) (std::ostream &stream, [mtk::UniStgGrid3D](#) &in)

Variables

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

16.1.1 Function Documentation

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

1. Print approximating coefficients for the interior.

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

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

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

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

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

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

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

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

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

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

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

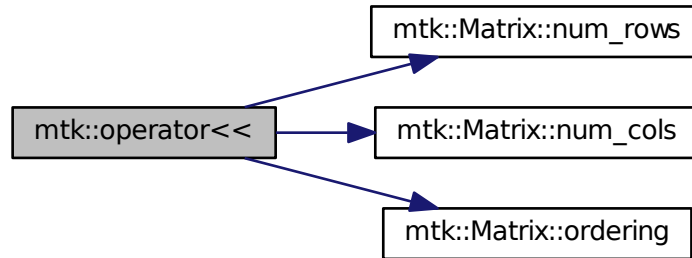
1. Print order of accuracy.
2. Print approximating coefficients for the interior.
3. No weights, thus print the mimetic boundary coefficients.

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

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

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

Here is the call graph for this function:



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

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

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

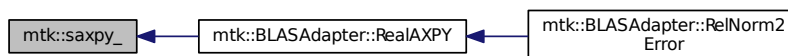
16.1.1.8 `std::ostream& mtk::operator<< (std::ostream & stream, mtk::Div1D & in)`

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

Definition at line 84 of file [mtk_div_1d.cc](#).

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

Here is the caller graph for this function:



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

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

The following options are provided:

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

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

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

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

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

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

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

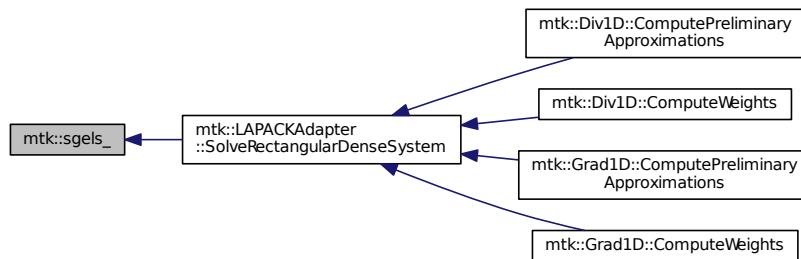
See also

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

Parameters

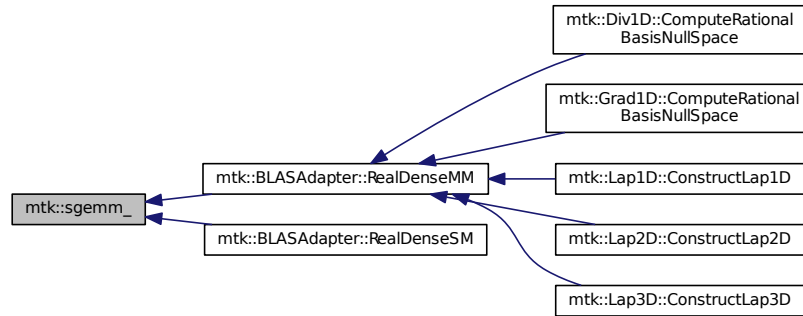
| | | |
|--------|--------------|------------------------------------------------------------------------|
| in | <i>trans</i> | Am I giving the transpose of the matrix? |
| in | <i>m</i> | The number of rows of the matrix a. $m \geq 0$. |
| in | <i>n</i> | The number of columns of the matrix a. $n \geq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

Here is the caller graph for this function:



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

Here is the caller graph for this function:



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

Here is the caller graph for this function:



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

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

SIDE = 'L' SIDE = 'R'

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

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

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

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

See also

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

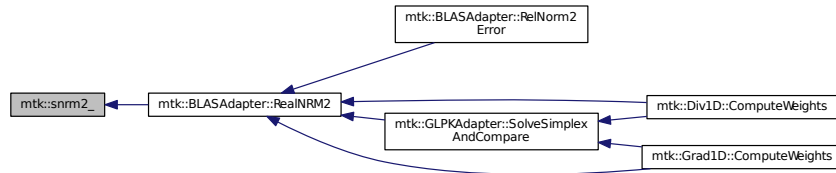
Parameters

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

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

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

Here is the caller graph for this function:



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

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

SIDE = 'L' SIDE = 'R'

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

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

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

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

See also

http://www.netlib.org/lapack/explore-html/d0/d98/sormqr_8f_source.html

Parameters

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

Chapter 17

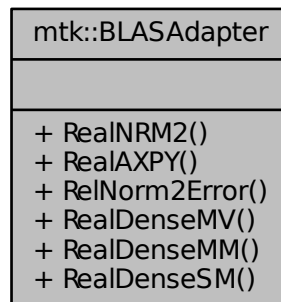
Class Documentation

17.1 mtk::BLASAdapter Class Reference

Adapter class for the BLAS API.

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

Collaboration diagram for mtk::BLASAdapter:



Static Public Member Functions

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

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

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

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

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

17.1.1 Detailed Description

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

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

See also

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

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

17.1.2 Member Function Documentation

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

Performs

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

Parameters

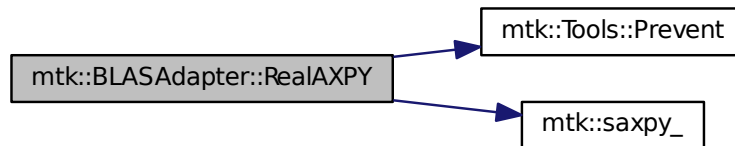
| | | |
|-----------------|------------------------|------------------------------|
| <code>in</code> | <code>alpha</code> | Scalar of the first array. |
| <code>in</code> | <code>xx</code> | First array. |
| <code>in</code> | <code>yy</code> | Second array. |
| <code>in</code> | <code>in_length</code>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

Returns

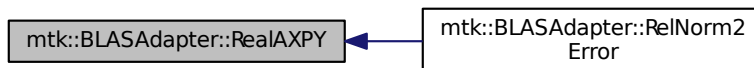
Norm-2 of the given array.

Definition at line 342 of file [mtk_blas_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Performs:

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

Parameters

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

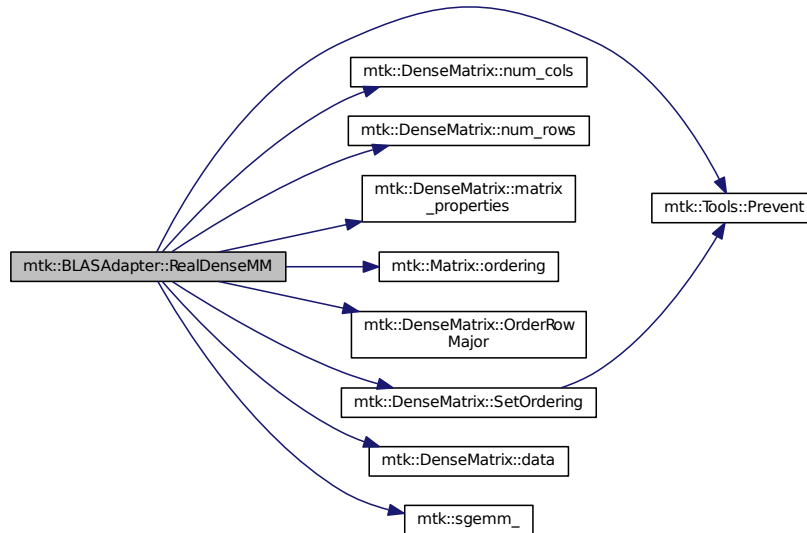
See also

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

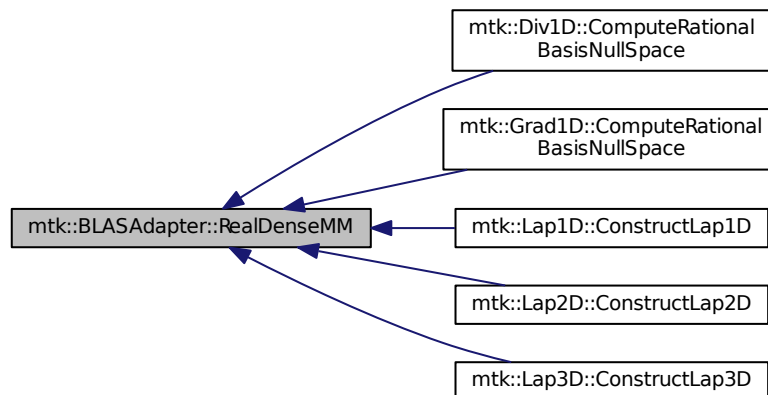
1. Make sure input matrices are row-major ordered.
2. Setup the problem.
3. Perform multiplication.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Performs

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

Parameters

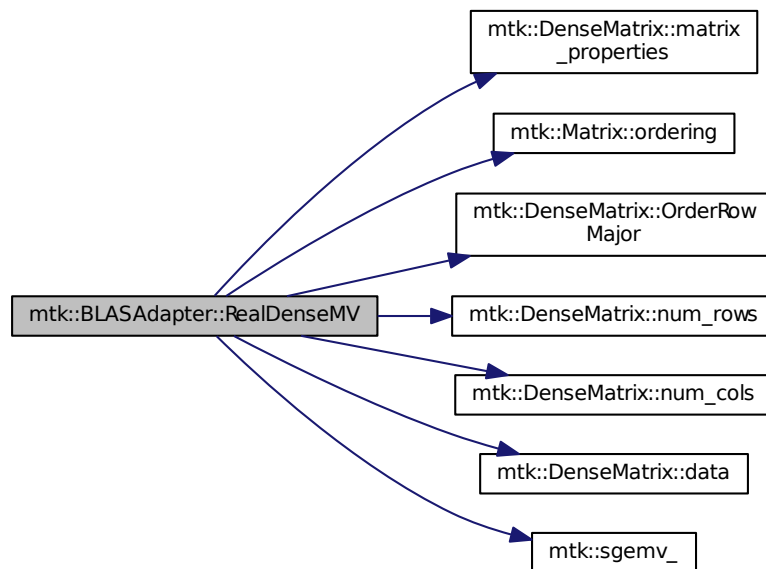
| | | |
|---------|--------------|-------------------------|
| in | <i>alpha</i> | First scalar. |
| in | <i>aa</i> | Given matrix. |
| in | <i>xx</i> | First vector. |
| in | <i>beta</i> | Second scalar. |
| in, out | <i>yy</i> | Second vector (output). |

See also

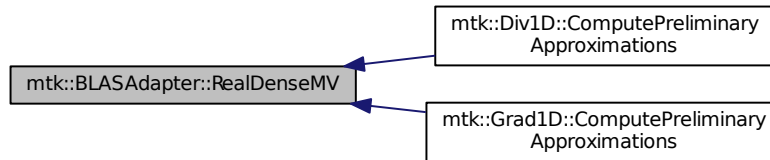
<http://ejspeiro.github.io/Netlib-and-CPP/>

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Performs:

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

Parameters

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

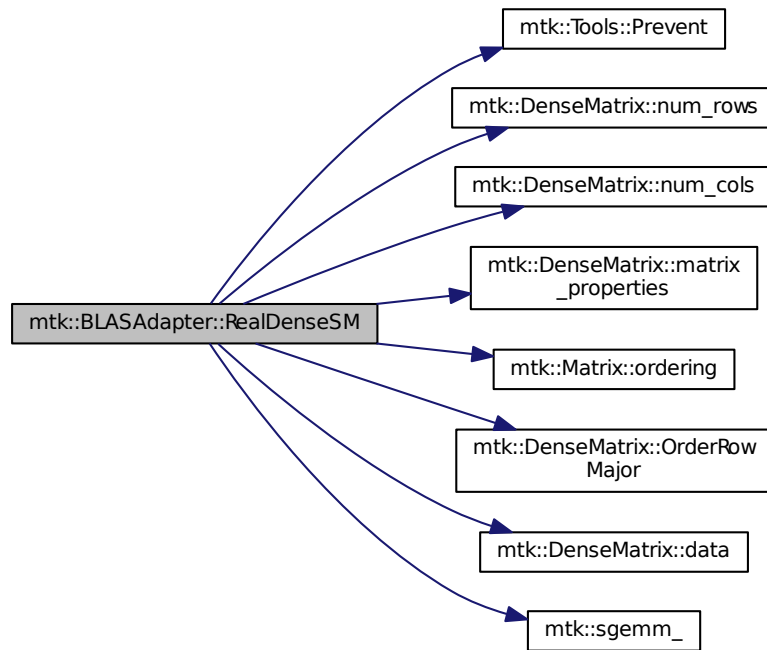
See also

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

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

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

Here is the call graph for this function:



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

Parameters

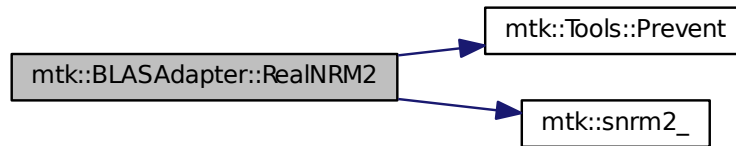
| | | |
|-----------------|------------------------|----------------------|
| <code>in</code> | <code>in</code> | Input array. |
| <code>in</code> | <code>in_length</code> | Length of the array. |

Returns

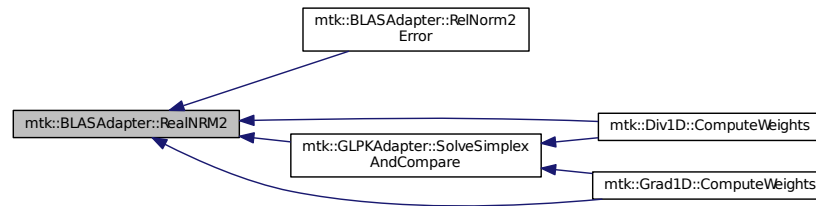
Norm-2 of the given array.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

We compute

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

Parameters

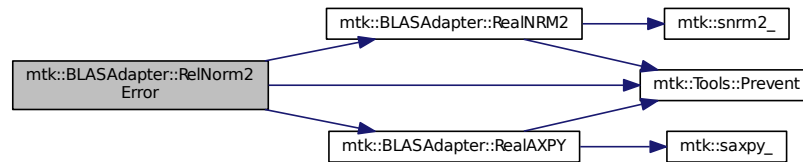
| | | |
|-----------|-----------------|------------------------------------------------------|
| <i>in</i> | <i>known</i> | Array containing the computed solution. |
| <i>in</i> | <i>computed</i> | Array containing the known solution (ref. solution). |

Returns

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

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

Here is the call graph for this function:



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

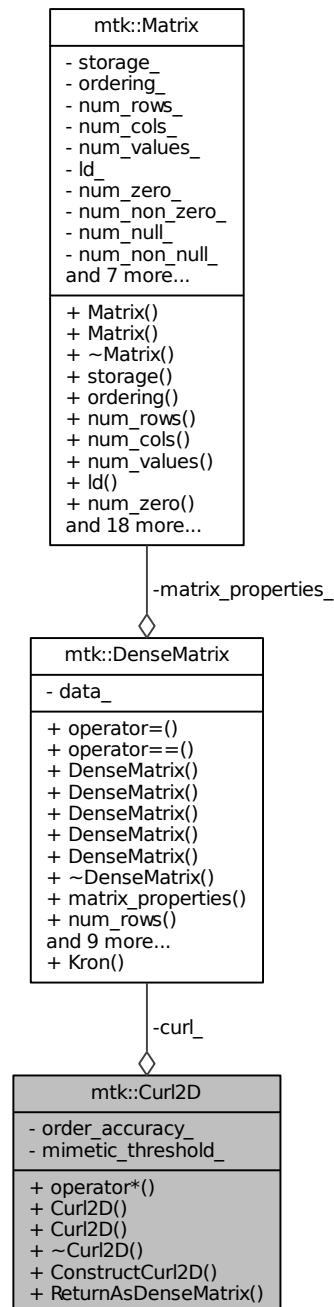
- [include/mtk_blas_adapter.h](#)
- [src/mtk_blas_adapter.cc](#)

17.2 mtk::Curl2D Class Reference

Implements a 2D mimetic curl operator.

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

Collaboration diagram for `mtk::Curl2D`:



Public Member Functions

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

Operator application operator on a grid.

- [Curl2D](#) ()

Default constructor.

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

Copy constructor.

- [~Curl2D](#) ()

Destructor.

- bool [ConstructCurl2D](#) (const [UniStgGrid2D](#) &grid, int order_accuracy=kDefaultOrderAccuracy, [Real](#) mimetic_↔ threshold=kDefaultMimeticThreshold)

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) () const

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) curl_

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

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

Mimetic Threshold.

17.2.1 Detailed Description

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

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

17.2.2 Constructor & Destructor Documentation

17.2.2.1 mtk::Curl2D::Curl2D ()

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

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

Parameters

| | | |
|--------------------|----------------------|-------------|
| in | curl | Given curl. |
|--------------------|----------------------|-------------|

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

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

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

17.2.3 Member Function Documentation

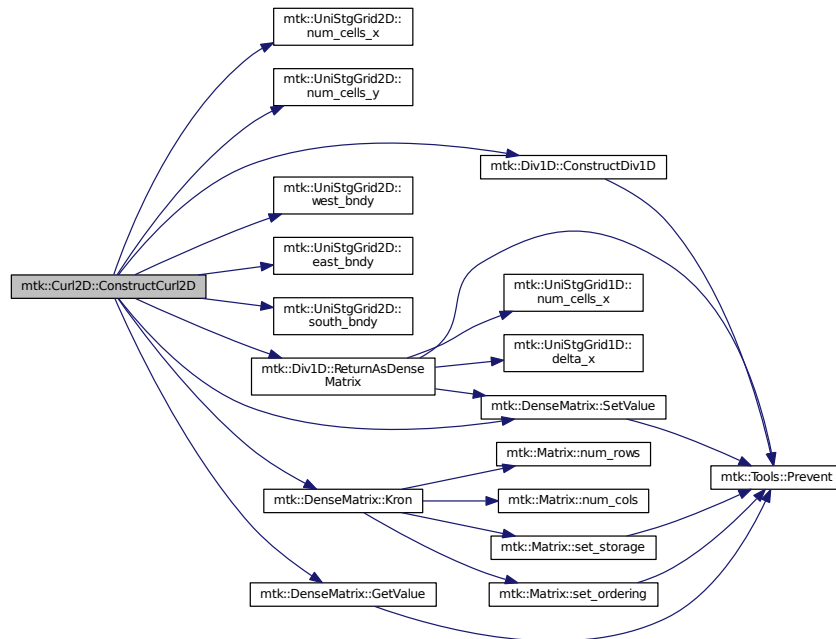
17.2.3.1 `bool mtk::Curl2D::ConstructCurl2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

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

Here is the call graph for this function:



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

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

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

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

Returns

The operator as a dense matrix.

Definition at line 157 of file [mtk_curl_2d.cc](#).

17.2.4 Member Data Documentation

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

Definition at line 112 of file [mtk_curl_2d.h](#).

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

Definition at line 116 of file [mtk_curl_2d.h](#).

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

Definition at line 114 of file [mtk_curl_2d.h](#).

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

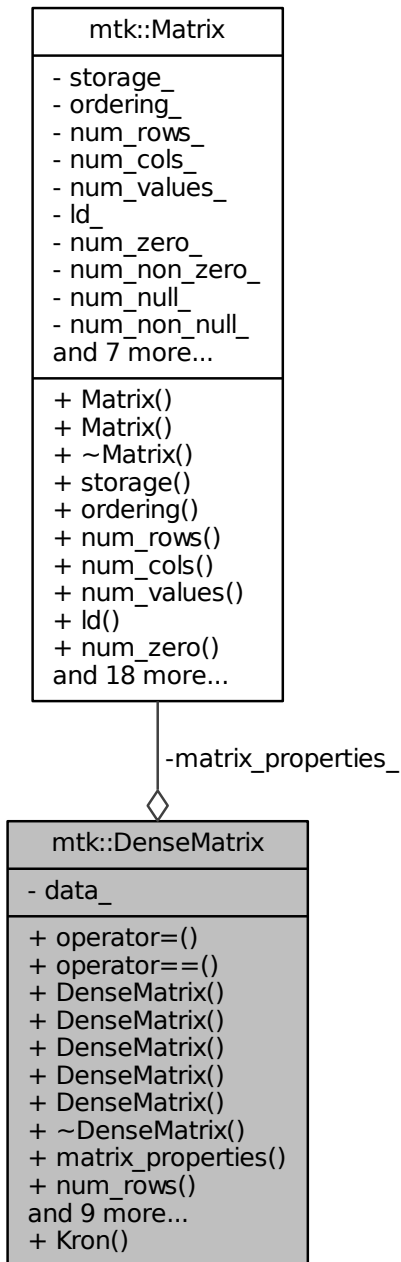
- [include/mtk_curl_2d.h](#)
- [src/mtk_curl_2d.cc](#)

17.3 mtk::DenseMatrix Class Reference

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

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

Collaboration diagram for mtk::DenseMatrix:



Public Member Functions

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

Overloaded assignment operator.

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

Static Public Member Functions

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

Private Attributes

- [Matrix](#) `matrix_properties_`

Data related to the matrix nature.

- [Real](#) * `data_`

Array holding the data in contiguous position in memory.

Friends

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

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

17.3.1 Detailed Description

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

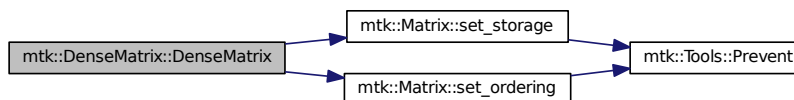
Definition at line 92 of file [mtk_dense_matrix.h](#).

17.3.2 Constructor & Destructor Documentation

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

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

Here is the call graph for this function:



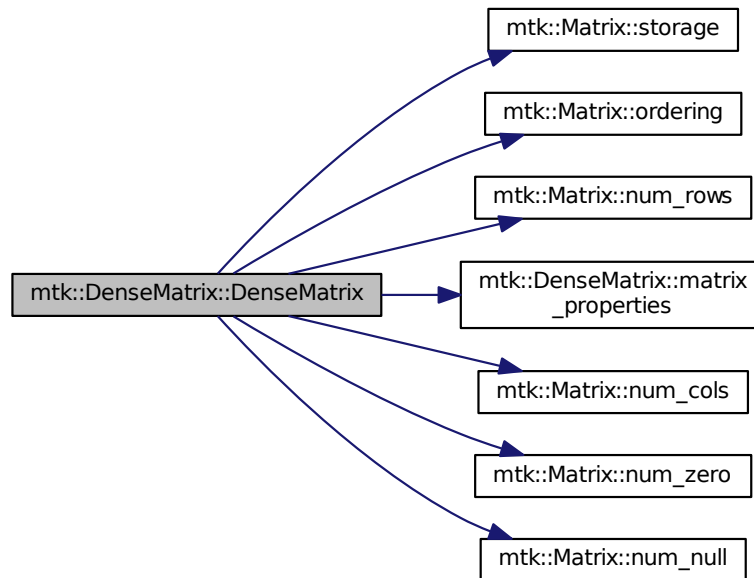
17.3.2.2 `mtk::DenseMatrix::DenseMatrix (const DenseMatrix &in)`

Parameters

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

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

Here is the call graph for this function:



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

Parameters

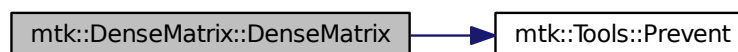
| | | |
|----|-----------------|----------------------------------------|
| in | <i>num_rows</i> | Number of rows of the required matrix. |
| in | <i>num_cols</i> | Number of rows of the required matrix. |

Exceptions

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

Definition at line 206 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



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

Used in the construction of the mimetic operators.

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

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

Parameters

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

Exceptions

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

Definition at line 228 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



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

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

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

This constructor generates a Vandermonde matrix, as defined above.

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

Parameters

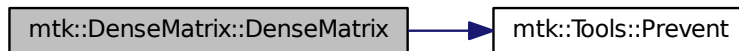
| | | |
|----|-------------------|------------------------------------------|
| in | <i>gen</i> | Given generator vector. |
| in | <i>gen_length</i> | Length generator vector. |
| in | <i>pro_length</i> | Length the progression. |
| in | <i>transpose</i> | Should the transpose be created instead? |

Exceptions

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

Definition at line 269 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



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

Definition at line 317 of file [mtk_dense_matrix.cc](#).

17.3.3 Member Function Documentation

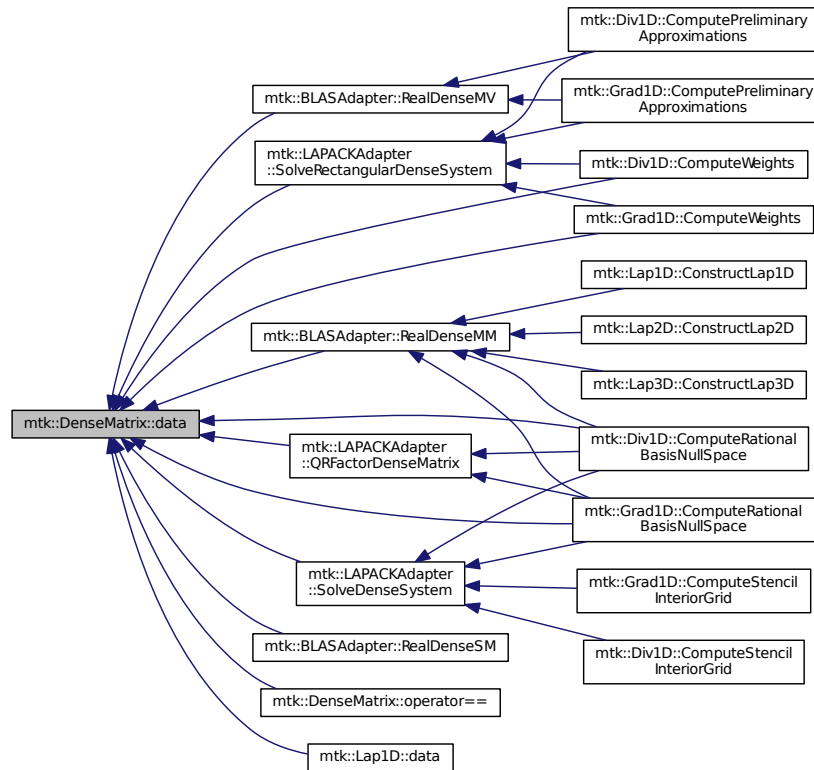
17.3.3.1 mtk::Real * mtk::DenseMatrix::data () const [noexcept]

Returns

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

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

Here is the caller graph for this function:



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

Parameters

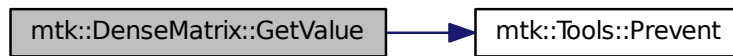
| | | |
|----|------------------|--------------------|
| in | <i>row_coord</i> | Row coordinate. |
| in | <i>col_coord</i> | Column coordinate. |

Returns

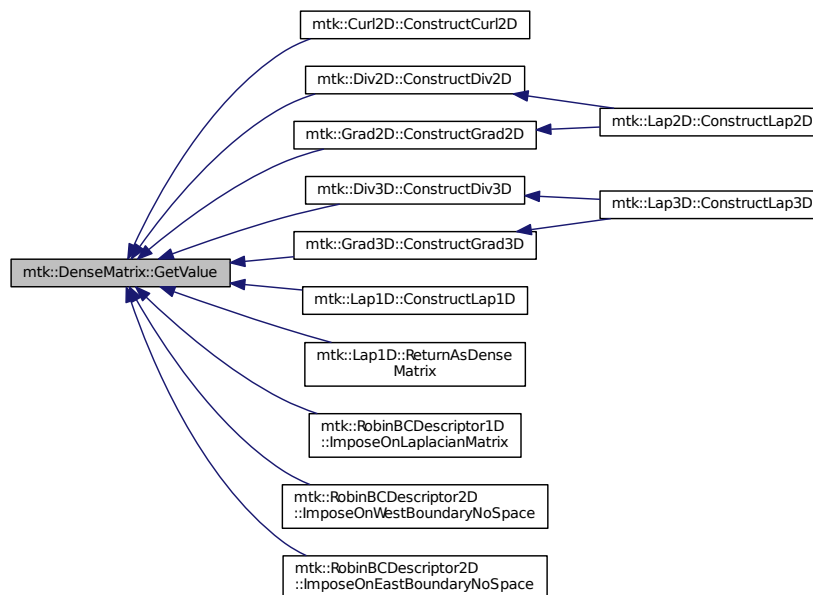
The required value at the specified coordinates.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

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

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

Exceptions

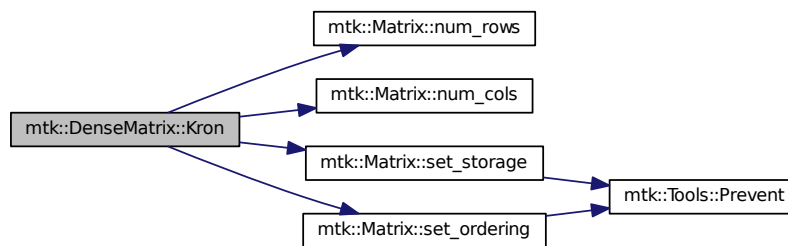
| |
|-----------------------------|
| <code>std::bad_alloc</code> |
|-----------------------------|

Todo Implement Kronecker product using the BLAS.

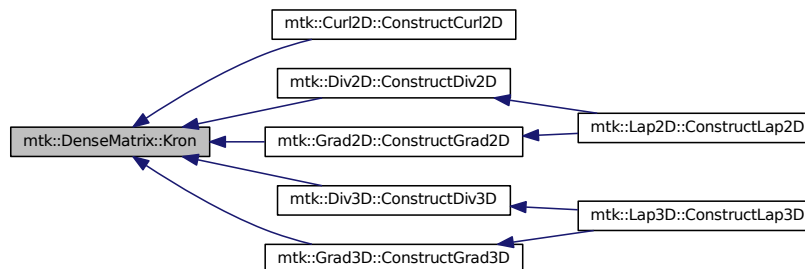
Todo Implement Kron using the BLAS.

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

Here is the call graph for this function:



Here is the caller graph for this function:



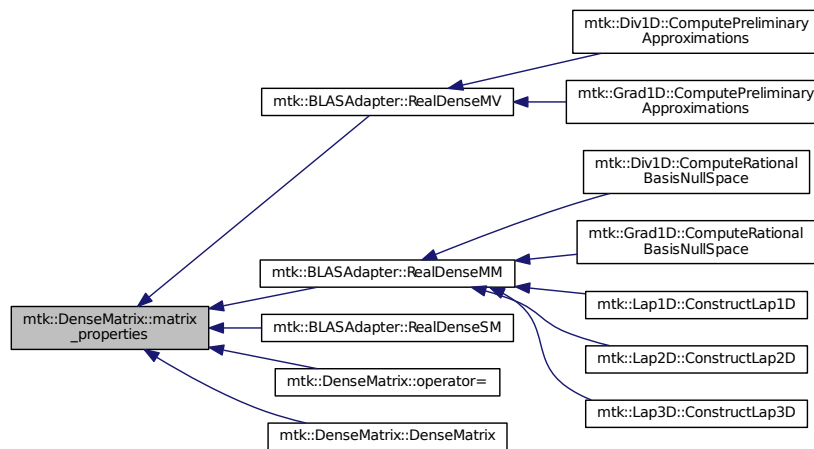
17.3.3.4 `mtk::Matrix mtk::DenseMatrix::matrix_properties () const` `[noexcept]`

Returns

Pointer to a [Matrix](#).

Definition at line 323 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



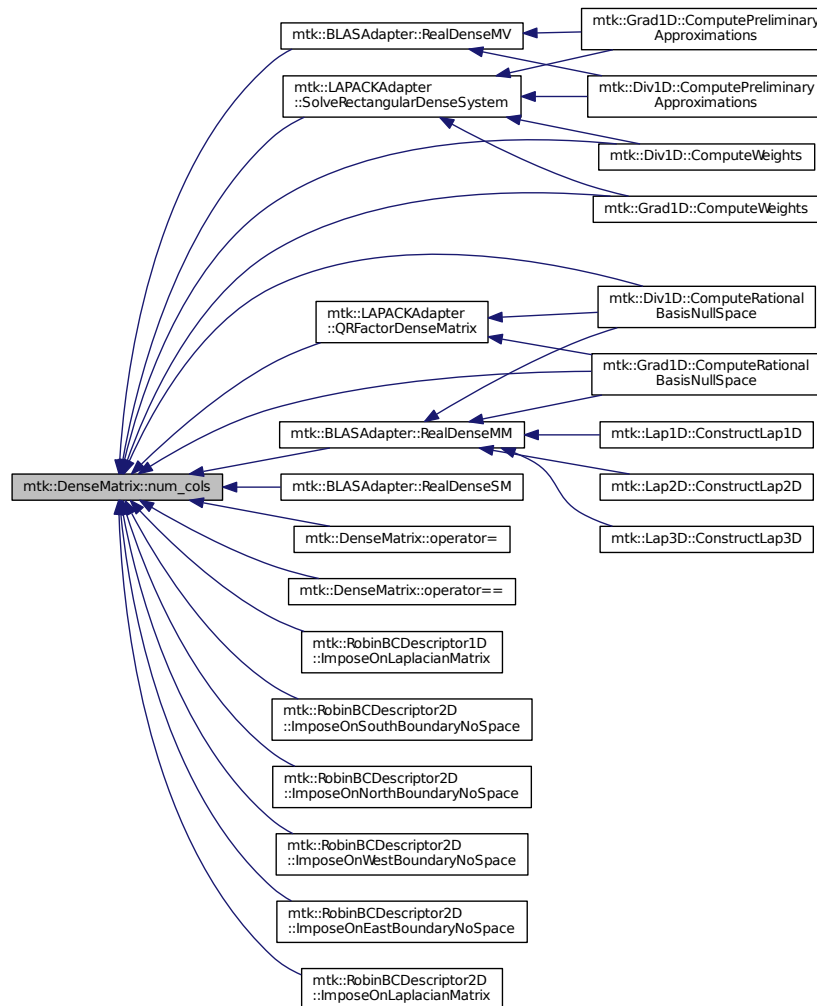
17.3.3.5 int mtk::DenseMatrix::num_cols () const [noexcept]

Returns

Number of columns of the matrix.

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

Here is the caller graph for this function:



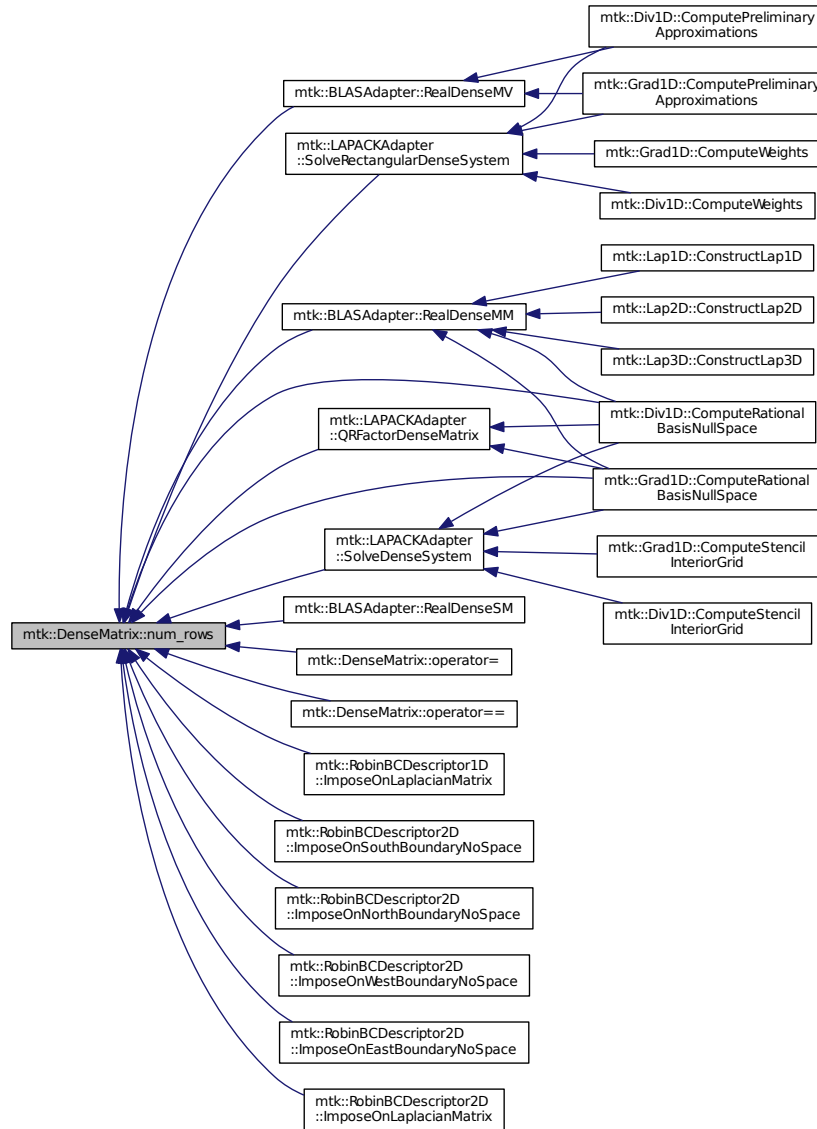
17.3.3.6 `int mtk::DenseMatrix::num_rows () const [noexcept]`

Returns

Number of rows of the matrix.

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

Here is the caller graph for this function:



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

Parameters

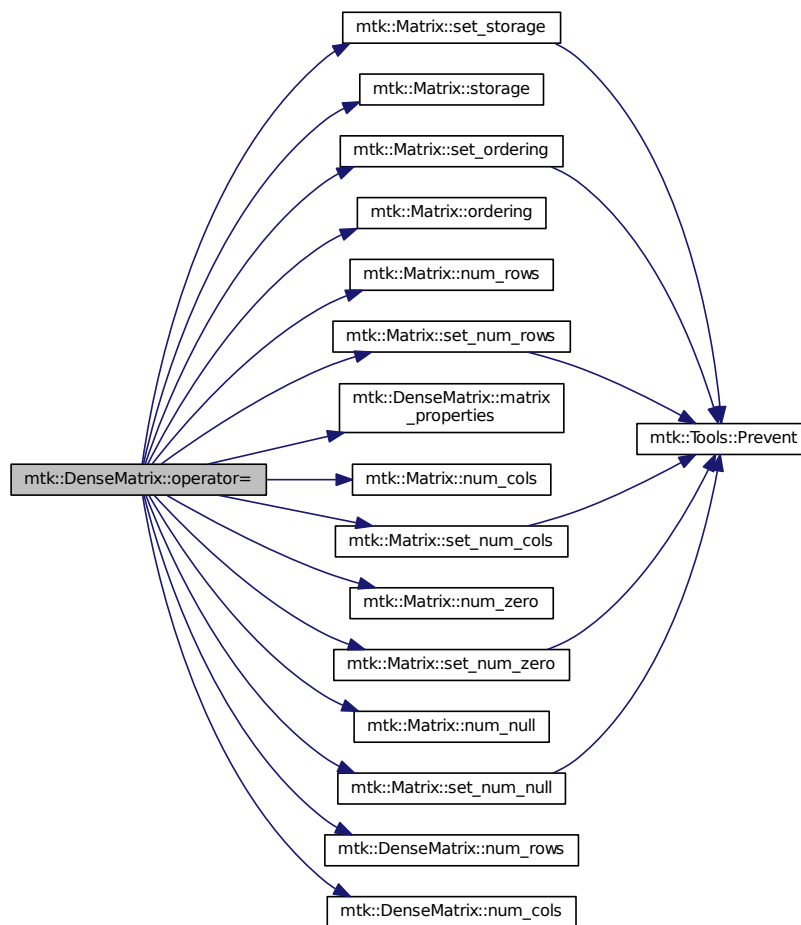
| | | |
|-----------|-----------|---------------|
| <i>in</i> | <i>in</i> | Given matrix. |
|-----------|-----------|---------------|

Returns

Copy of the given matrix.

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

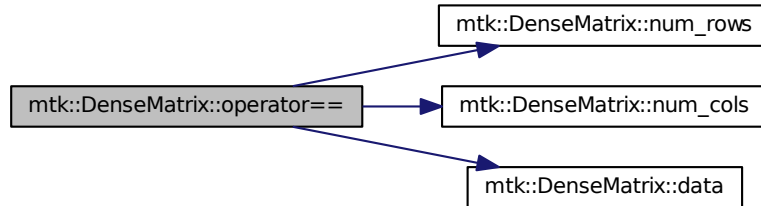
Here is the call graph for this function:



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

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

Here is the call graph for this function:

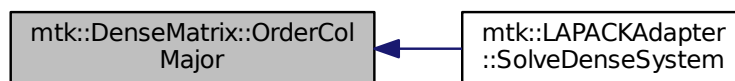


17.3.3.9 `void mtk::DenseMatrix::OrderColMajor ()`

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

Definition at line 457 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:

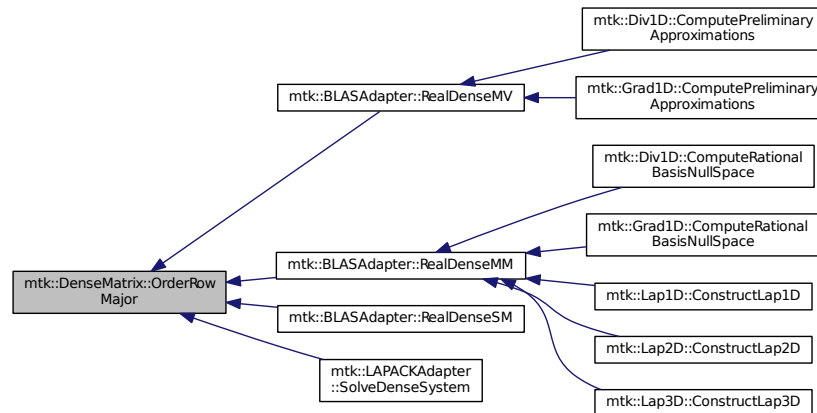


17.3.3.10 `void mtk::DenseMatrix::OrderRowMajor ()`

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

Definition at line 416 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



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

Parameters

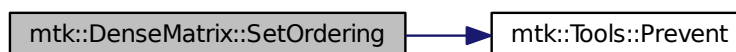
| | | |
|----|----|-----------|
| in | oo | Ordering. |
|----|----|-----------|

Returns

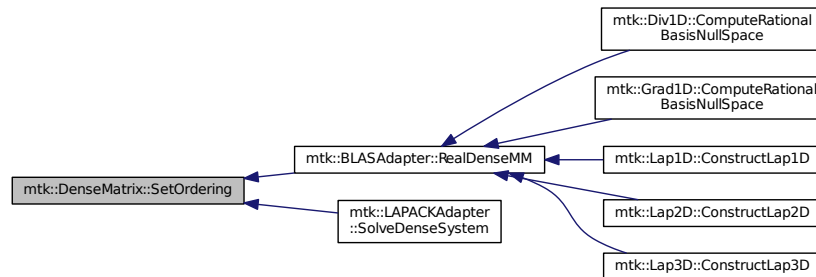
The required value at the specified coordinates.

Definition at line 328 of file [mtk_dense_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



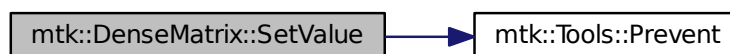
17.3.3.12 `void mtk::DenseMatrix::SetValue (const int & row_coord, const int & col_coord, const Real & val)` [noexcept]

Parameters

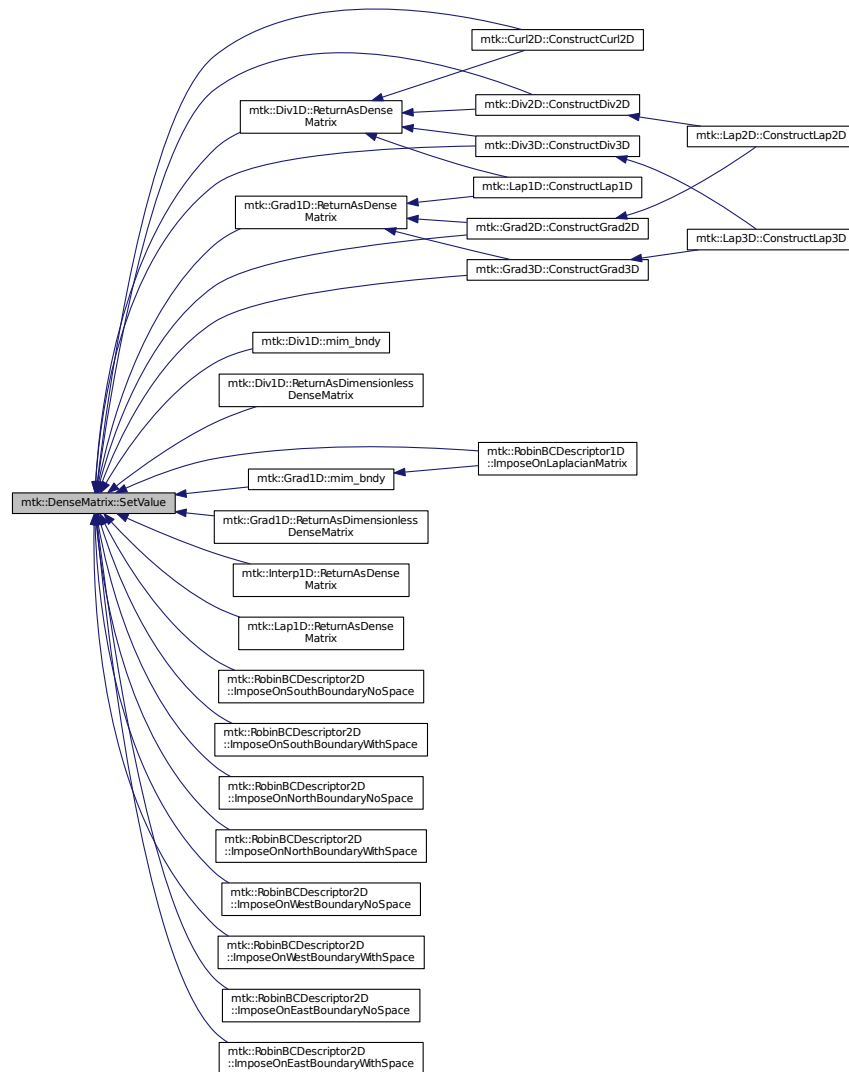
| | | |
|----|------------------|----------------------------------|
| in | <i>row_coord</i> | Row coordinate. |
| in | <i>col_coord</i> | Column coordinate. |
| in | <i>val</i> | Row Actual value to be inserted. |

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

Here is the call graph for this function:



Here is the caller graph for this function:

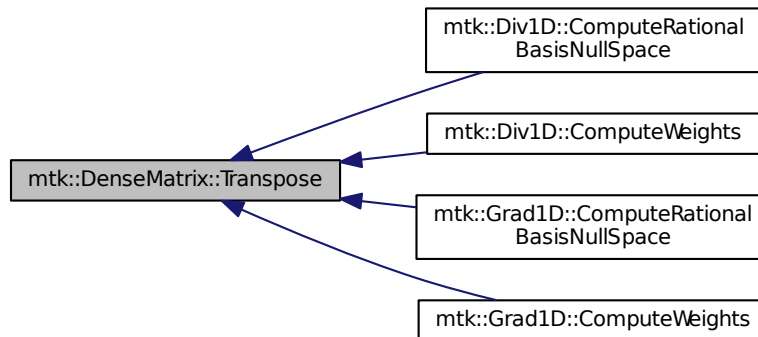


17.3.3.13 void mtk::DenseMatrix::Transpose ()

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

Definition at line 379 of file [mtk_dense_matrix.cc](#).

Here is the caller graph for this function:



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

Parameters

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

Returns

Success of the file writing process.

See also

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

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

17.3.4 Friends And Related Function Documentation

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

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

17.3.5 Member Data Documentation

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

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

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

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

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

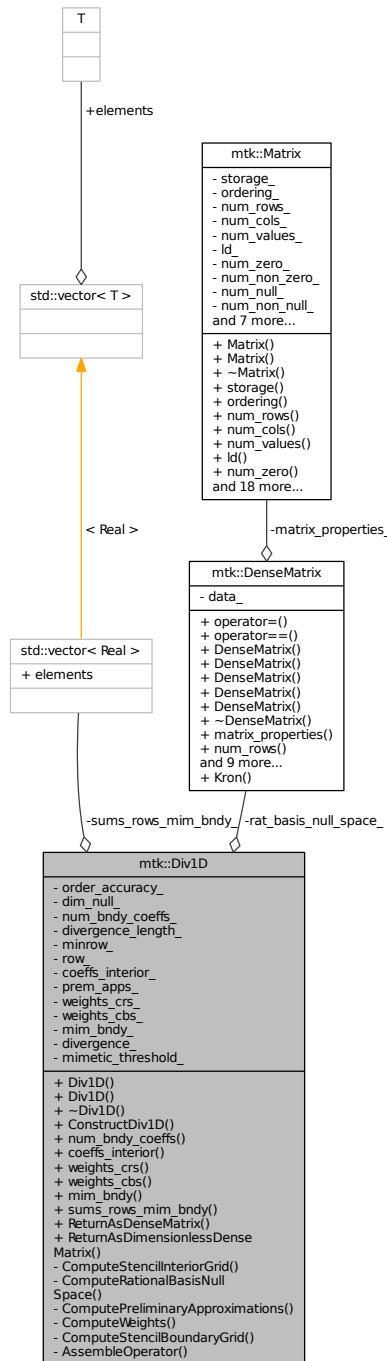
- [include/mtk_dense_matrix.h](#)
- [src/mtk_dense_matrix.cc](#)

17.4 `mtk::Div1D` Class Reference

Implements a 1D mimetic divergence operator.

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

Collaboration diagram for mtk::Div1D:



Public Member Functions

- [Div1D\(\)](#)

- Default constructor.*

 - [Div1D](#) (const [Div1D](#) &div)
- Copy constructor.*

 - [~Div1D](#) ()
- Destructor.*

 - bool [ConstructDiv1D](#) (int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_threshold=[kDefaultMimeticThreshold](#))
- Factory method implementing the CBS Algorithm to build operator.*

 - int [num_bndy_coefs](#) () const
- Returns how many coefficients are approximating at the boundary.*

 - [Real](#) * [coefs_interior](#) () const
- Returns coefficients for the interior of the grid.*

 - [Real](#) * [weights_crs](#) (void) const
- Return collection of weights as computed by the CRSA.*

 - [Real](#) * [weights_cbs](#) (void) const
- Return collection of weights as computed by the CBSA.*

 - [DenseMatrix](#) [mim_bndy](#) () const
- Return collection of mimetic approximations at the boundary.*

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

 - [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid) const
- Return the operator as a dense matrix.*

 - [DenseMatrix](#) [ReturnAsDimensionlessDenseMatrix](#) (int num_cells_x) const
- Returns the operator as a dimensionless dense matrix.*

Private Member Functions

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

Private Attributes

- int [order_accuracy_](#)
- Order of numerical accuracy of the operator.*
- int [dim_null_](#)
- Dim. null-space for boundary approximations.*
- int [num_bndy_coefs_](#)

- Req. coeffs. per bndy pt. uni. order accuracy.*
- int [divergence_length_](#)
Length of the output array.
- int [minrow_](#)
Row from the optimizer with the minimum rel. nor.
- int [row_](#)
Row currently processed by the optimizer.
- [DenseMatrix](#) [rat_basis_null_space_](#)
Rational b. null-space w. bndy.
- [Real](#) * [coeffs_interior_](#)
Interior stencil.
- [Real](#) * [prem_apps_](#)
2D array of boundary preliminary approximations.
- [Real](#) * [weights_crs_](#)
Array containing weights from CRSA.
- [Real](#) * [weights_cbs_](#)
Array containing weights from CBSA.
- [Real](#) * [mim_bndy_](#)
Array containing mimetic boundary approximations.
- [Real](#) * [divergence_](#)
Output array containing the operator and weights.
- std::vector< [Real](#) > [sums_rows_mim_bndy_](#)
Sum of each mimetic boundary row.
- [Real](#) [mimetic_threshold_](#)
< Mimetic threshold.

Friends

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

17.4.1 Detailed Description

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

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

17.4.2 Constructor & Destructor Documentation

17.4.2.1 mtk::Div1D::Div1D ()

Definition at line 136 of file [mtk_div_1d.cc](#).

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

Parameters

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

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

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

Definition at line 168 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 1483 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 333 of file [mtk_div_1d.cc](#).

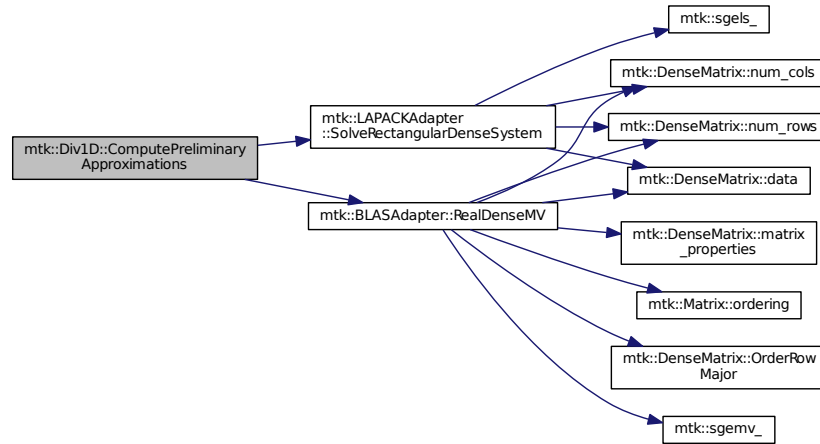
17.4.3.3 bool mtk::Div1D::ComputePreliminaryApproximations (void) [private]

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

Here is the call graph for this function:



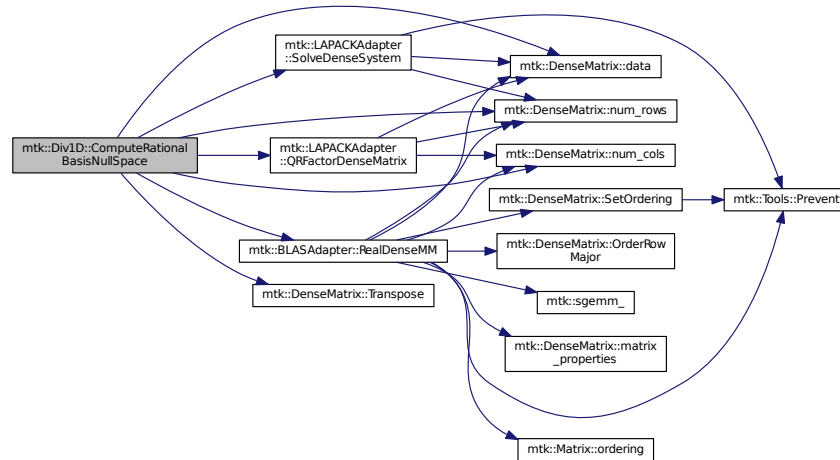
17.4.3.4 `bool mtk::Div1D::ComputeRationalBasisNullSpace (void) [private]`

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 595 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 1364 of file [mtk_div_1d.cc](#).

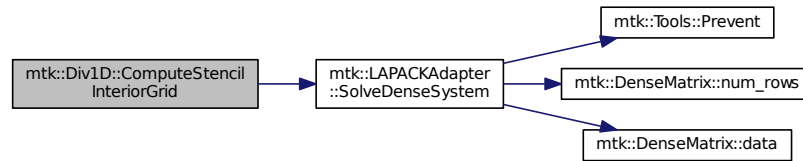
17.4.3.6 `bool mtk::Div1D::ComputeStencilInteriorGrid (void) [private]`

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

Here is the call graph for this function:



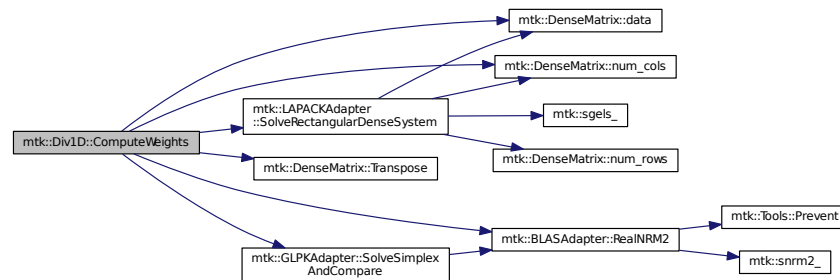
17.4.3.7 bool mtk::Div1D::ComputeWeights (void) [private]

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

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

Definition at line 991 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.

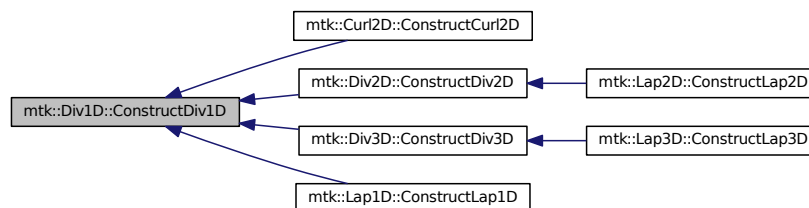
1. Compute stencil for the interior cells.
2. Compute a rational basis for the null-space for the first matrix.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 189 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



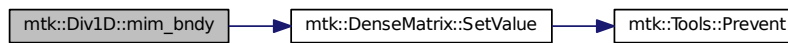
17.4.3.9 `mtk::DenseMatrix mtk::Div1D::mim_bndy () const`

Returns

Collection of mimetic approximations at the boundary.

Definition at line 348 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



17.4.3.10 int mtk::Div1D::num_bndy_coeffs () const

Returns

How many coefficients are approximating at the boundary.

Definition at line 328 of file [mtk_div_1d.cc](#).

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

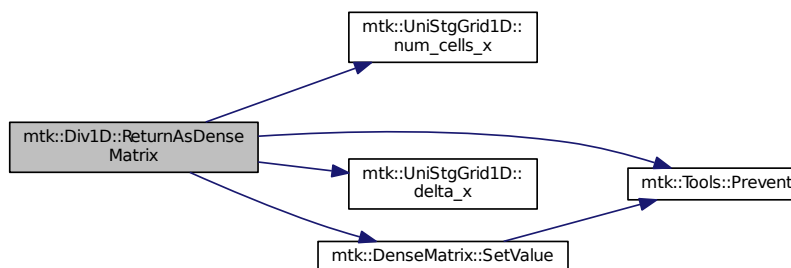
Returns

The operator as a dense matrix.

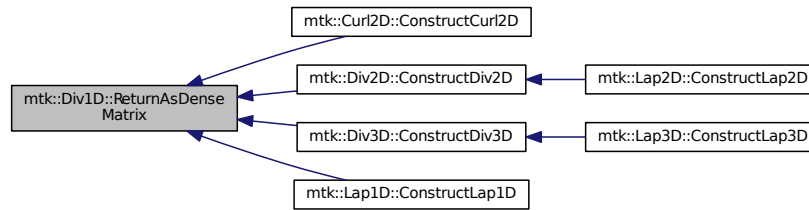
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 368 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

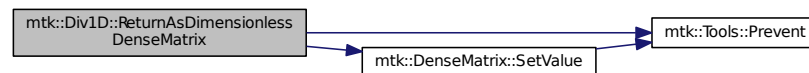
Returns

The operator as a dimensionless dense matrix.

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

Definition at line 432 of file [mtk_div_1d.cc](#).

Here is the call graph for this function:



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

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 363 of file [mtk_div_1d.cc](#).

17.4.3.14 `mtk::Real * mtk::Div1D::weights_cbs (void) const`

Returns

Collection of weights as computed by the CBSA.

Definition at line 343 of file [mtk_div_1d.cc](#).

17.4.3.15 `mtk::Real * mtk::Div1D::weights_crs (void) const`

Returns

Collection of weights as computed by the CRSA.

Definition at line 338 of file [mtk_div_1d.cc](#).

17.4.4 Friends And Related Function Documentation

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

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

Definition at line 84 of file [mtk_div_1d.cc](#).

17.4.5 Member Data Documentation

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

Definition at line 218 of file [mtk_div_1d.h](#).

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

Definition at line 210 of file [mtk_div_1d.h](#).

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

Definition at line 223 of file [mtk_div_1d.h](#).

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

Definition at line 212 of file [mtk_div_1d.h](#).

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

Definition at line 222 of file [mtk_div_1d.h](#).

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

Definition at line 227 of file [mtk_div_1d.h](#).

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

Definition at line 213 of file [mtk_div_1d.h](#).

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

Definition at line 211 of file [mtk_div_1d.h](#).

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

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

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

Definition at line 219 of file [mtk_div_1d.h](#).

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

Definition at line 216 of file [mtk_div_1d.h](#).

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

Definition at line 214 of file [mtk_div_1d.h](#).

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

Definition at line 225 of file [mtk_div_1d.h](#).

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

Definition at line 221 of file [mtk_div_1d.h](#).

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

Definition at line 220 of file [mtk_div_1d.h](#).

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

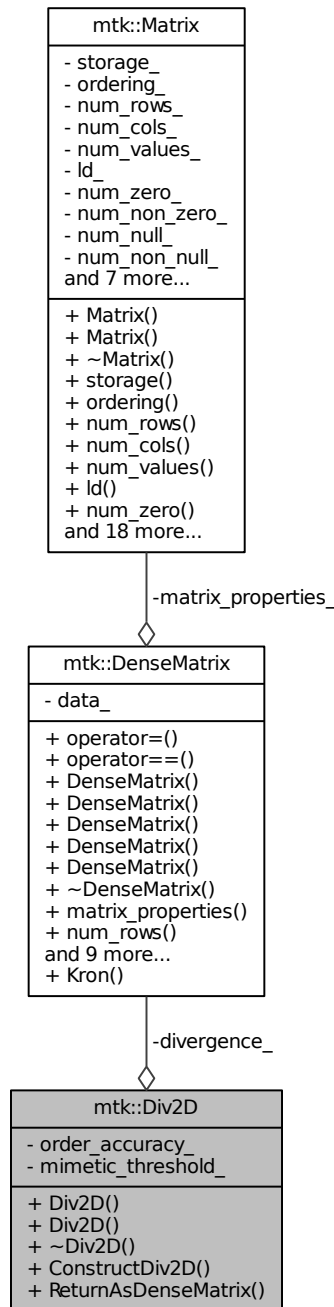
- [include/mtk_div_1d.h](#)
- [src/mtk_div_1d.cc](#)

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_ \leftrightarrow threshold=[kDefaultMimeticThreshold](#))

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) () const

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) divergence_

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

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

Mimetic Threshold.

17.5.1 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

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

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

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

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

17.5.3 Member Function Documentation

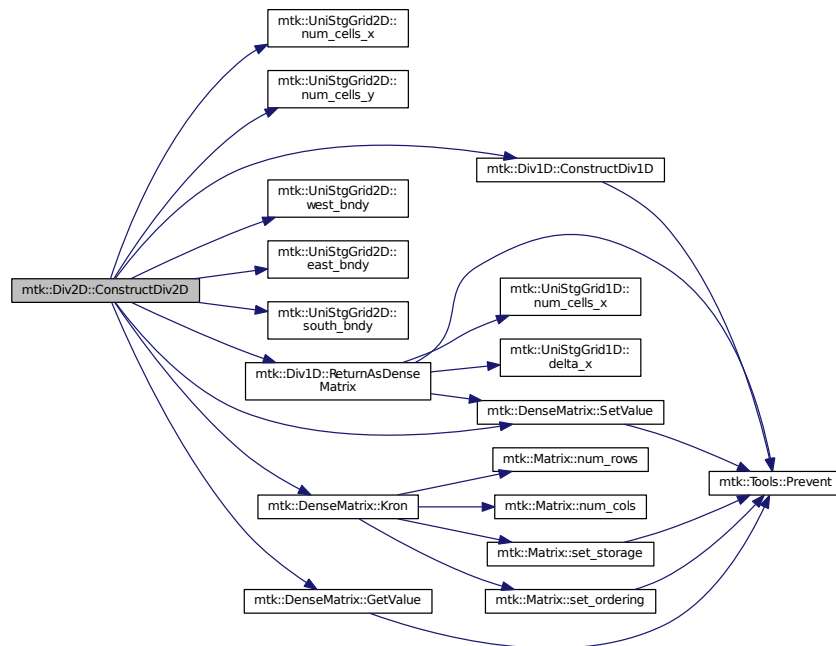
17.5.3.1 `bool mtk::Div2D::ConstructDiv2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Returns

The operator as a dense matrix.

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

Here is the caller graph for this function:

**17.5.4 Member Data Documentation****17.5.4.1 DenseMatrix mtk::Div2D::divergence_ [private]**

Definition at line 108 of file [mtk_div_2d.h](#).

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

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

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

Definition at line 110 of file [mtk_div_2d.h](#).

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

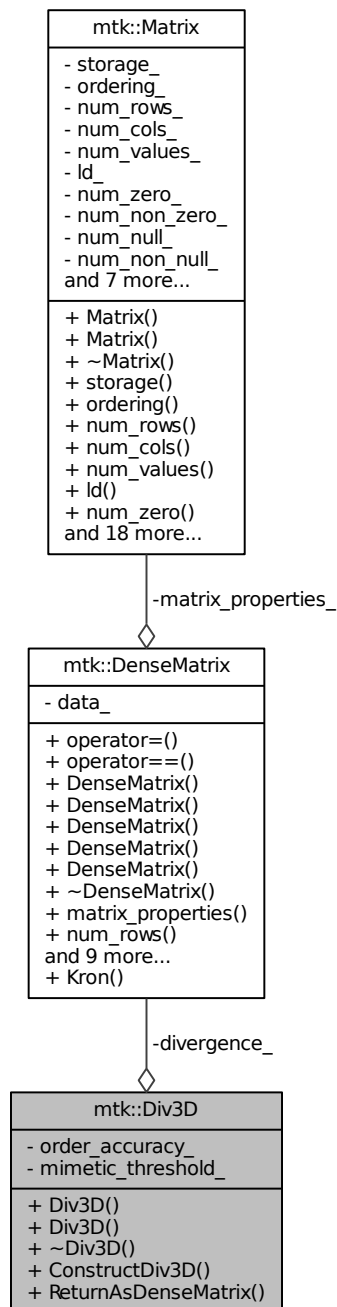
- [include/mtk_div_2d.h](#)
- [src/mtk_div_2d.cc](#)

17.6 mtk::Div3D Class Reference

Implements a 3D mimetic divergence operator.

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

Collaboration diagram for mtk::Div3D:



Public Member Functions

- [Div3D \(\)](#)

Default constructor.

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

Copy constructor.

- [~Div3D](#) ()

Destructor.

- bool [ConstructDiv3D](#) (const [UniStgGrid3D](#) &grid, int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_↔ 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

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

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

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

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

17.6.3 Member Function Documentation

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

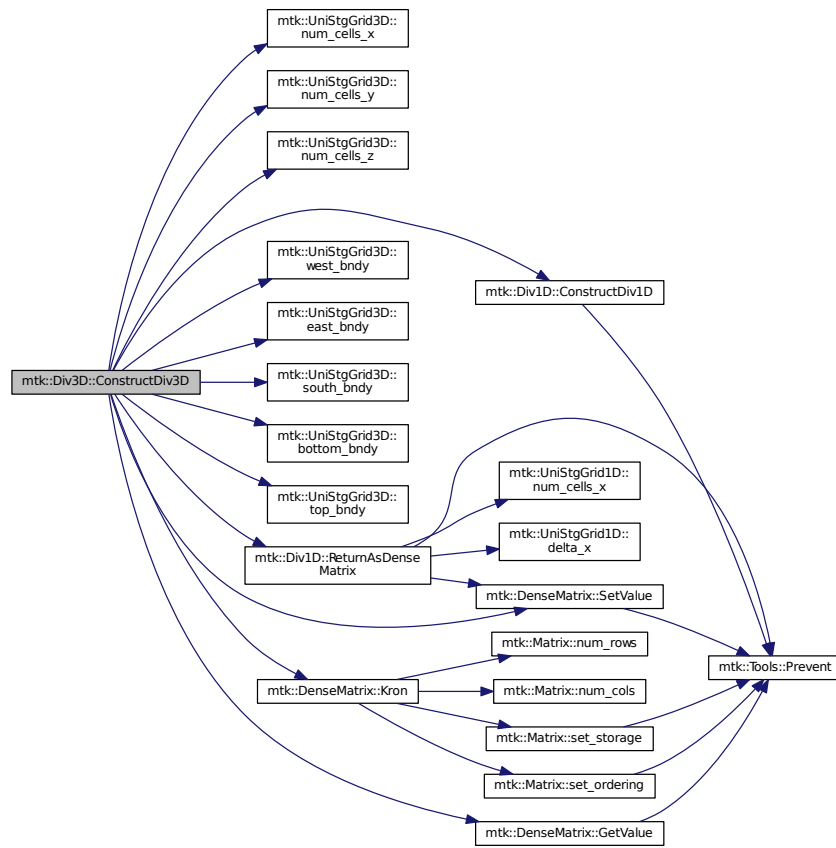
Returns

Success of the construction.

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

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Returns

The operator as a dense matrix.

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

Here is the caller graph for this function:



17.6.4 Member Data Documentation

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

Definition at line 108 of file [mtk_div_3d.h](#).

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

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

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

Definition at line 110 of file [mtk_div_3d.h](#).

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

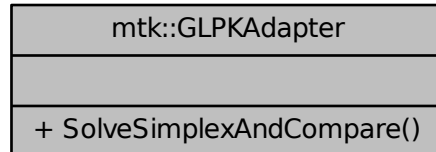
- [include/mtk_div_3d.h](#)
- [src/mtk_div_3d.cc](#)

17.7 mtk::GLPKAdapter Class Reference

Adapter class for the GLPK API.

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

Collaboration diagram for mtk::GLPKAdapter:



Static Public Member Functions

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

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

17.7.1 Detailed Description

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

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

Warning

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

See also

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

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

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

17.7.2 Member Function Documentation

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

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

Parameters

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

Returns

Relative error computed between attained solution and provided ref.

Warning

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

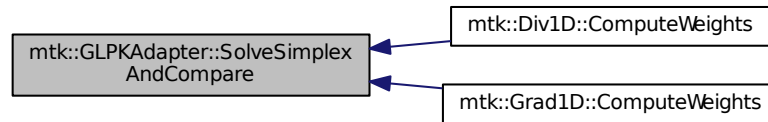
1. Memory allocation.
2. Fill the problem.
3. Copy the row to the vector objective.
4. Forming the RHS.
5. Setting up the objective function.
6. Setting up constraints.
7. Copy the matrix minus the row objective to the glpk problem.
8. Solve problem.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

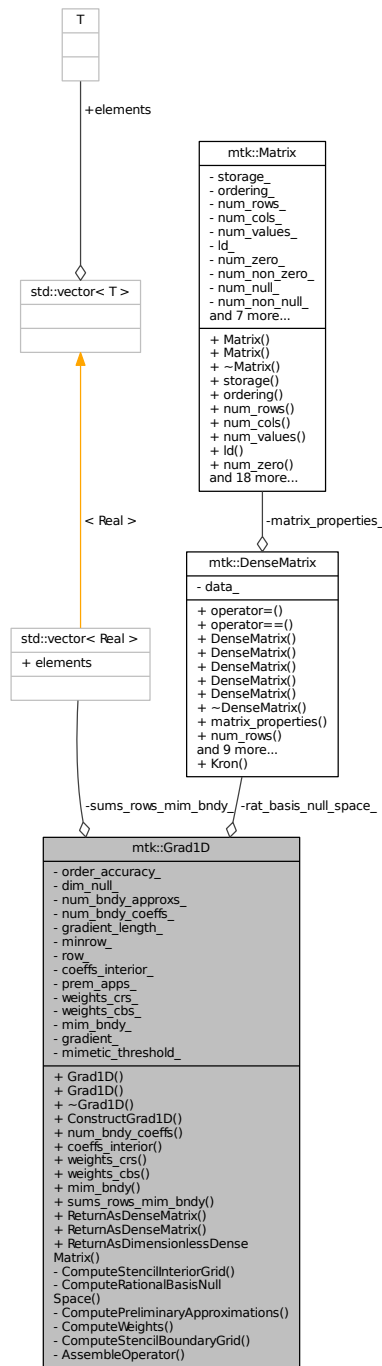
- [include/mtk_glpk_adapter.h](#)
- [src/mtk_glpk_adapter.cc](#)

17.8 mtk::Grad1D Class Reference

Implements a 1D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad1D:



Public Member Functions

- [Grad1D \(\)](#)

Default constructor.

- [Grad1D](#) (const [Grad1D](#) &grad)

Copy constructor.

- [~Grad1D](#) ()

Destructor.

- bool [ConstructGrad1D](#) (int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_threshold=[kDefaultMimeticThreshold](#))

Factory method implementing the CBS Algorithm to build operator.

- int [num_bndy_coefs](#) () const

Returns how many coefficients are approximating at the boundary.

- [Real](#) * [coefs_interior](#) () const

Returns coefficients for the interior of the grid.

- [Real](#) * [weights_crs](#) (void) const

Returns collection of weights as computed by the CRSA.

- [Real](#) * [weights_cbs](#) (void) const

Returns collection of weights as computed by the CBSA.

- [DenseMatrix](#) [mim_bndy](#) () const

Return collection of mimetic approximations at the boundary.

- std::vector< [Real](#) > [sums_rows_mim_bndy](#) () const

Return collection of row-sums mimetic approximations at the boundary.

- [DenseMatrix](#) [ReturnAsDenseMatrix](#) ([Real](#) west, [Real](#) east, int num_cells_x) const

Returns the operator as a dense matrix.

- [DenseMatrix](#) [ReturnAsDenseMatrix](#) (const [UniStgGrid1D](#) &grid) const

Returns the operator as a dense matrix.

- [DenseMatrix](#) [ReturnAsDimensionlessDenseMatrix](#) (int num_cells_x) const

Returns the operator as a dimensionless dense matrix.

Private Member Functions

- bool [ComputeStencilInteriorGrid](#) (void)

Stage 1 of the CBS Algorithm.

- bool [ComputeRationalBasisNullSpace](#) (void)

Stage 2.1 of the CBS Algorithm.

- bool [ComputePreliminaryApproximations](#) (void)

Stage 2.2 of the CBS Algorithm.

- bool [ComputeWeights](#) (void)

Stage 2.3 of the CBS Algorithm.

- bool [ComputeStencilBoundaryGrid](#) (void)

Stage 2.4 of the CBS Algorithm.

- bool [AssembleOperator](#) (void)

Stage 3 of the CBS Algorithm.

Private Attributes

- int [order_accuracy_](#)
Order of numerical accuracy of the operator.
- int [dim_null_](#)
Dim. null-space for boundary approximations.
- int [num_bndy_approxs_](#)
Req. approximations at and near the boundary.
- int [num_bndy_coeffs_](#)
Req. coeffs. per bndy pt. uni. order accuracy.
- int [gradient_length_](#)
Length of the output array.
- int [minrow_](#)
Row from the optimizer with the minimum rel. nor.
- int [row_](#)
Row currently processed by the optimizer.
- [DenseMatrix](#) [rat_basis_null_space_](#)
Rational b. null-space w. bndy.
- [Real](#) * [coeffs_interior_](#)
Interior stencil.
- [Real](#) * [prem_apps_](#)
2D array of boundary preliminary approximations.
- [Real](#) * [weights_crs_](#)
Array containing weights from CRSA.
- [Real](#) * [weights_cbs_](#)
Array containing weights from CBSA.
- [Real](#) * [mim_bndy_](#)
Array containing mimetic boundary approximations.
- [Real](#) * [gradient_](#)
Output array containing the operator and weights.
- [std::vector](#)< [Real](#) > [sums_rows_mim_bndy_](#)
Sum of each mimetic boundary row.
- [Real](#) [mimetic_threshold_](#)
< Mimetic threshold.

Friends

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

17.8.1 Detailed Description

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

Definition at line 83 of file [mtk_grad_1d.h](#).

17.8.2 Constructor & Destructor Documentation

17.8.2.1 `mtk::Grad1D::Grad1D ()`

Definition at line 143 of file [mtk_grad_1d.cc](#).

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

Parameters

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

Definition at line 160 of file [mtk_grad_1d.cc](#).

17.8.2.3 `mtk::Grad1D::~~Grad1D ()`

Definition at line 177 of file [mtk_grad_1d.cc](#).

17.8.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

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

Definition at line 1581 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 342 of file [mtk_grad_1d.cc](#).

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

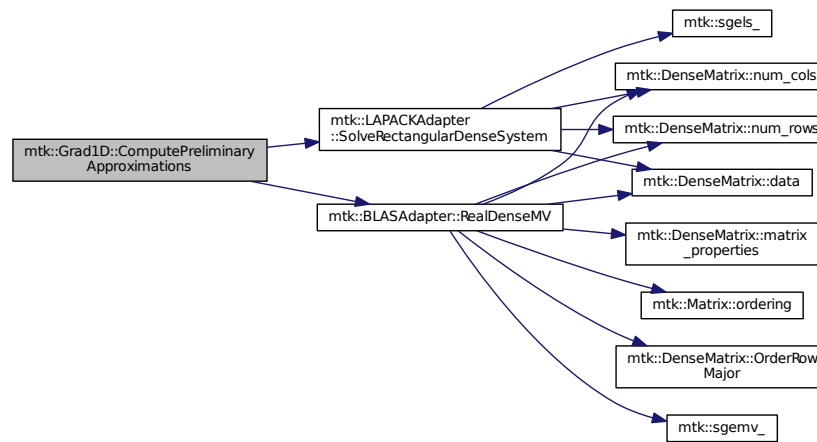
Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.
2. Compute the `dim_null` near-the-boundary columns of the pi matrix.
3. Create the Vandermonde matrix for this iteration.

4. New order-selector vector (gets re-written with LAPACK solutions).
5. Solving $TT*rr = ob$ yields the columns rr of the kk matrix.
6. Scale the kk matrix to make it a rational basis for null-space.
7. Extract the last dim_null values of the pre-scaled ob .
8. Once we possess the bottom elements, we proceed with the scaling.

Definition at line 852 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



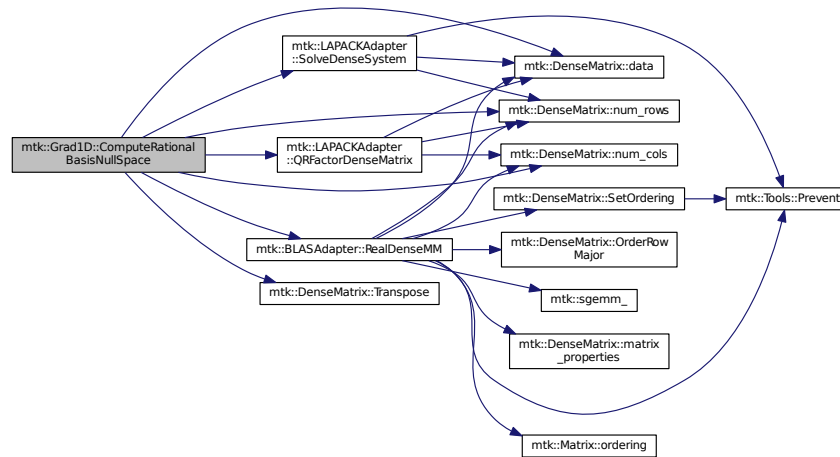
17.8.3.4 `bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) [private]`

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 669 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 the operator is mimetic.

Definition at line 1457 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 572 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



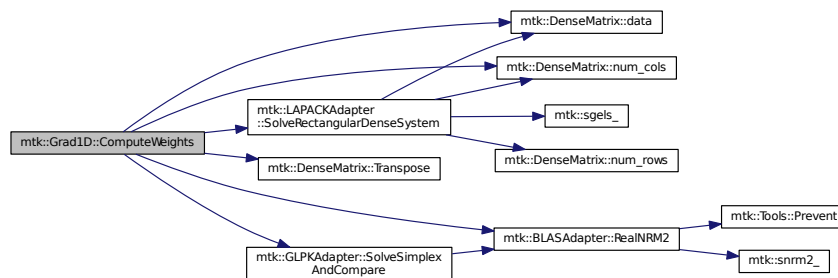
17.8.3.7 bool mtk::Grad1D::ComputeWeights (void) [private]

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

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

Definition at line 1073 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)

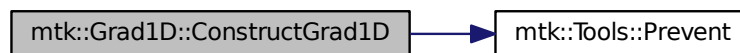
Returns

Success of the solution.

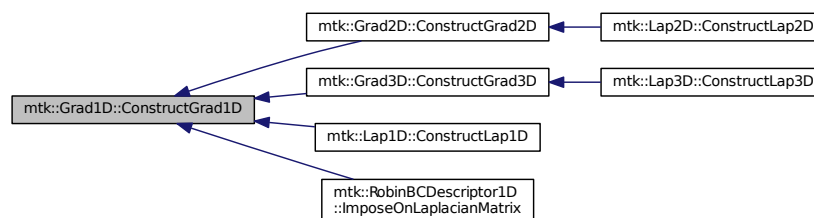
1. Compute stencil for the interior cells.
2. Compute a rational null-space from the first matrix transposed.
3. Compute preliminary approximation (non-mimetic) on the boundaries.
4. Compute quadrature weights to impose the mimetic conditions.
5. Compute real approximation (mimetic) on the boundaries.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

Definition at line 198 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



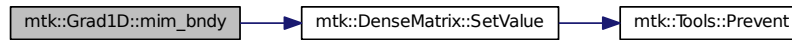
17.8.3.9 `mtk::DenseMatrix mtk::Grad1D::mim_bndy () const`

Returns

Collection of mimetic approximations at the boundary.

Definition at line 357 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Returns

How many coefficients are approximating at the boundary.

Definition at line 337 of file [mtk_grad_1d.cc](#).

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

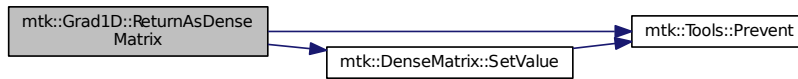
Returns

The operator as a dense matrix.

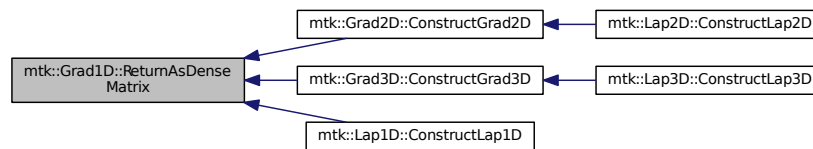
1. Insert mimetic boundary at the west.
2. Insert coefficients for the interior of the grid.
3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 377 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

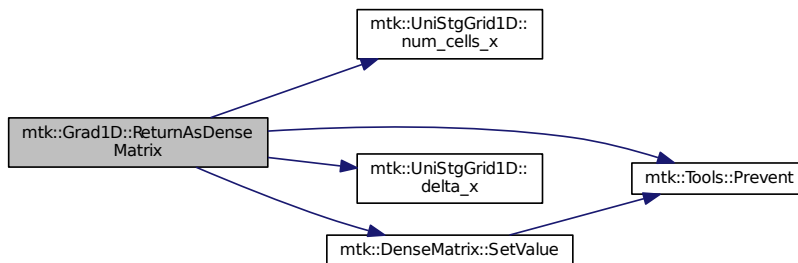
Returns

The operator as a dense matrix.

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

Definition at line 446 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



17.8.3.13 **mtk::DenseMatrix** mtk::Grad1D::ReturnAsDimensionlessDenseMatrix (int *num_cells_x*) const

Returns

The operator as a dimensionless dense matrix.

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

Definition at line 510 of file [mtk_grad_1d.cc](#).

Here is the call graph for this function:



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

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 372 of file [mtk_grad_1d.cc](#).

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

Returns

Collection of weights as computed by the CBSA.

Definition at line 352 of file [mtk_grad_1d.cc](#).

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

Returns

Success of the solution.

Definition at line 347 of file [mtk_grad_1d.cc](#).

17.8.4 Friends And Related Function Documentation

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

1. Print order of accuracy.

2. Print approximating coefficients for the interior.
3. Print mimetic weights.
4. Print mimetic approximations at the boundary.

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

17.8.5 Member Data Documentation

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

Definition at line 226 of file [mtk_grad_1d.h](#).

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

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

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

Definition at line 231 of file [mtk_grad_1d.h](#).

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

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

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

Definition at line 230 of file [mtk_grad_1d.h](#).

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

Definition at line 235 of file [mtk_grad_1d.h](#).

17.8.5.7 **int** `mtk::Grad1D::minrow_` `[private]`

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

17.8.5.8 **int** `mtk::Grad1D::num_bndy_approxs_` `[private]`

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

17.8.5.9 **int** `mtk::Grad1D::num_bndy_coeffs_` `[private]`

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

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

Definition at line 216 of file [mtk_grad_1d.h](#).

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

Definition at line 227 of file [mtk_grad_1d.h](#).

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

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

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

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

17.8.5.14 `std::vector<Real> mtk::Grad1D::sums_rows_mim_bndy_ [private]`

Definition at line 233 of file [mtk_grad_1d.h](#).

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

Definition at line 229 of file [mtk_grad_1d.h](#).

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

Definition at line 228 of file [mtk_grad_1d.h](#).

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

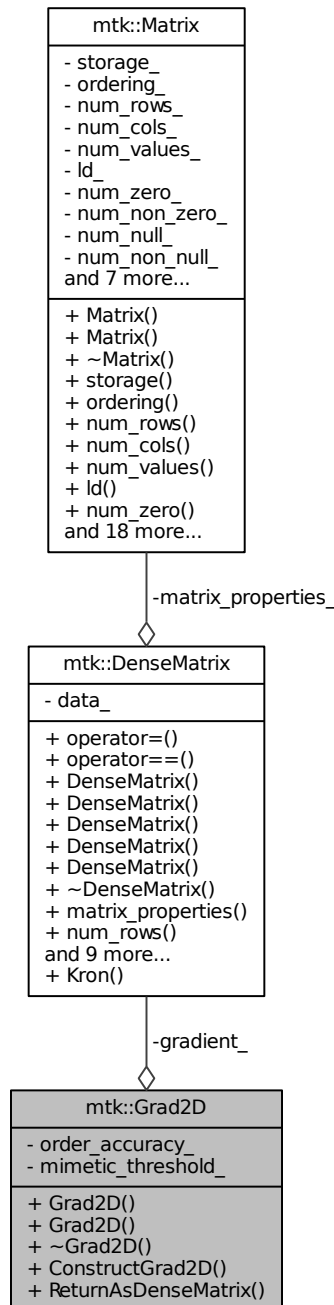
- [include/mtk_grad_1d.h](#)
- [src/mtk_grad_1d.cc](#)

17.9 mtk::Grad2D Class Reference

Implements a 2D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad2D:



Public Member Functions

- [Grad2D](#) ()

Default constructor.

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

Copy constructor.

- [~Grad2D](#) ()

Destructor.

- bool [ConstructGrad2D](#) (const [UniStgGrid2D](#) &grid, int order_accuracy=kDefaultOrderAccuracy, [Real](#) mimetic_↔ threshold=kDefaultMimeticThreshold)

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) () const

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) gradient_

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

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

Mimetic Threshold.

17.9.1 Detailed Description

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

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

17.9.2 Constructor & Destructor Documentation

17.9.2.1 mtk::Grad2D::Grad2D ()

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

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

Parameters

| | | |
|--------------------|---------------------|-------------------|
| 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

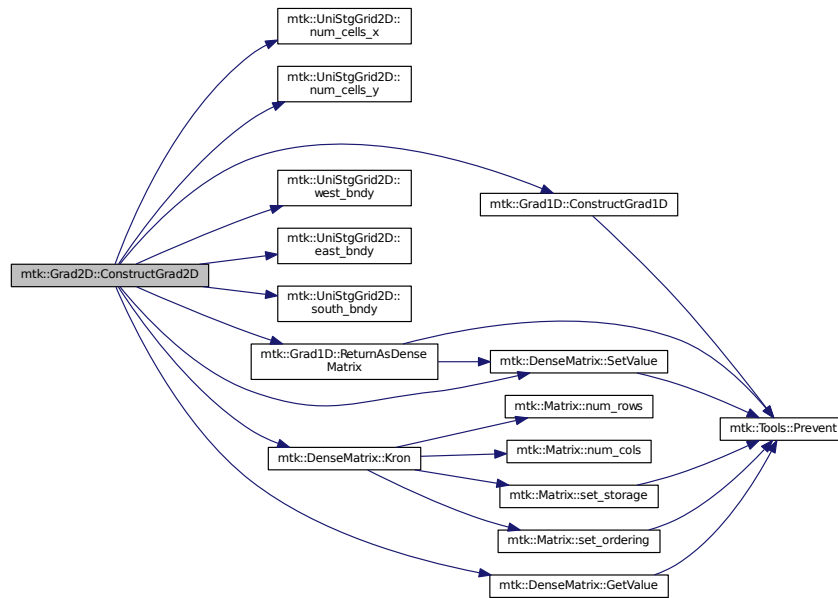
17.9.3.1 `bool mtk::Grad2D::ConstructGrad2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

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 108 of file [mtk_grad_2d.h](#).

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

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

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

Definition at line 110 of file [mtk_grad_2d.h](#).

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

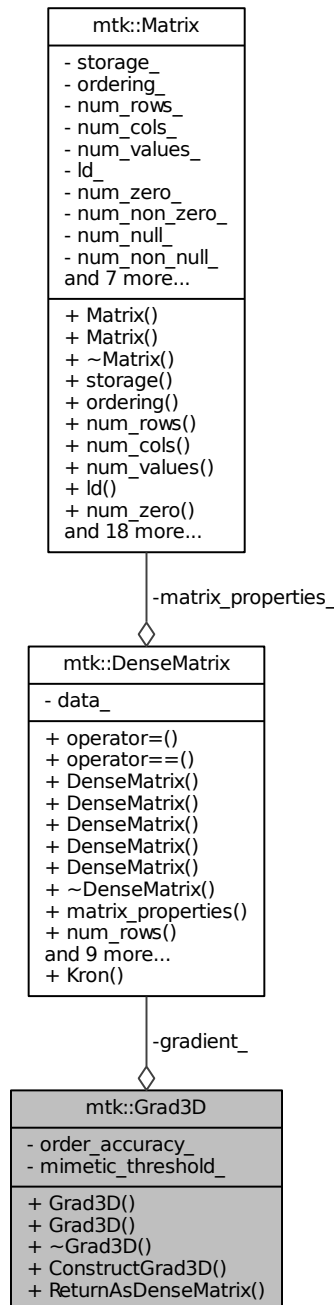
- [include/mtk_grad_2d.h](#)
- [src/mtk_grad_2d.cc](#)

17.10 mtk::Grad3D Class Reference

Implements a 3D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad3D:



Public Member Functions

- [Grad3D \(\)](#)

Default constructor.

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

Copy constructor.

- [~Grad3D](#) ()

Destructor.

- bool [ConstructGrad3D](#) (const [UniStgGrid3D](#) &grid, int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_↔ threshold=[kDefaultMimeticThreshold](#))

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) () const

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix](#) gradient_

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

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

Mimetic Threshold.

17.10.1 Detailed Description

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

Definition at line 76 of file [mtk_grad_3d.h](#).

17.10.2 Constructor & Destructor Documentation

17.10.2.1 mtk::Grad3D::Grad3D ()

Definition at line 67 of file [mtk_grad_3d.cc](#).

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

Parameters

| | | |
|--------------------|---------------------|-------------------|
| 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](#).

Here is the caller graph for this function:



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

Returns

The operator as a dense matrix.

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

Here is the caller graph for this function:



17.10.4 Member Data Documentation

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

Definition at line 108 of file [mtk_grad_3d.h](#).

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

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

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

Definition at line 110 of file [mtk_grad_3d.h](#).

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

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

17.11 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

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

Collaboration diagram for mtk::Interp1D:

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

Public Member Functions

- [Interp1D \(\)](#)
Default constructor.
- [Interp1D \(const \[Interp1D\]\(#\) &interp\)](#)
Copy constructor.
- [~Interp1D \(\)](#)
Destructor.
- [bool ConstructInterp1D \(int order_accuracy=kDefaultOrderAccuracy, mtk::DirInterp dir=mtk::DirInterp::SCALA↔
R_TO_VECTOR\)](#)
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

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

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

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

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

17.11.3 Member Function Documentation

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

Returns

Coefficients for the interior of the grid.

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

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

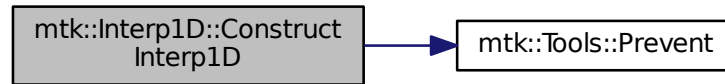
Returns

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:



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

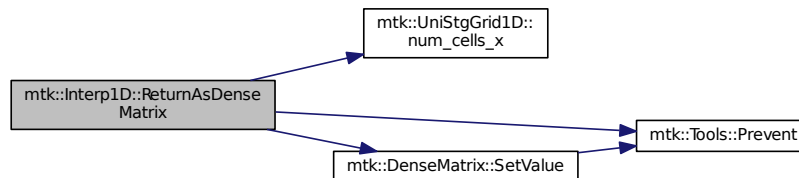
Returns

The operator as a dense matrix.

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

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

Here is the call graph for this function:



17.11.4 Friends And Related Function Documentation

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

1. Print approximating coefficients for the interior.

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

17.11.5 Member Data Documentation

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

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

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

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

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

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

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

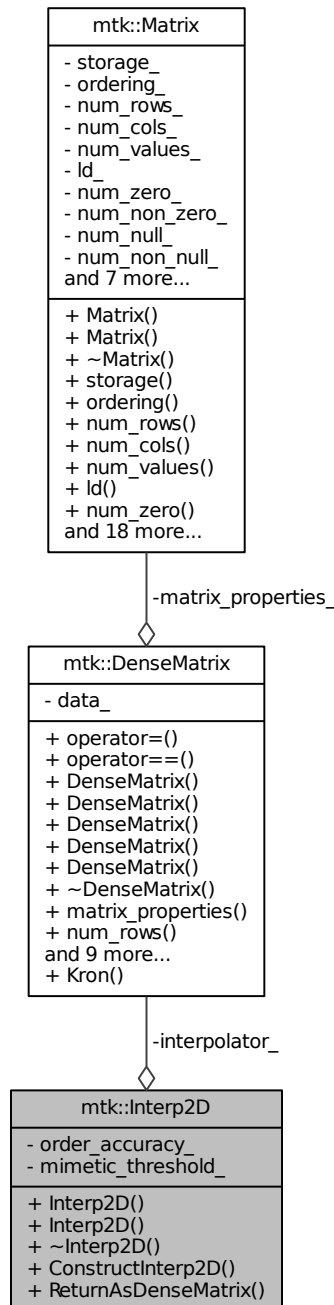
- [include/mtk_interp_1d.h](#)
- [src/mtk_interp_1d.cc](#)

17.12 mtk::Interp2D Class Reference

Implements a 2D interpolation operator.

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

Collaboration diagram for mtk::Interp2D:



Public Member Functions

- [Interp2D \(\)](#)

Default constructor.

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

Copy constructor.

- [~Interp2D](#) ()

Destructor.

- [DenseMatrix ConstructInterp2D](#) (const [UniStgGrid2D](#) &grid, int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_threshold=[kDefaultMimeticThreshold](#))

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) ()

Return the operator as a dense matrix.

Private Attributes

- [DenseMatrix interpolator_](#)

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

- [Real mimetic_threshold_](#)

Mimetic Threshold.

17.12.1 Detailed Description

This class implements a 2D interpolation operator.

Definition at line 76 of file [mtk_interp_2d.h](#).

17.12.2 Constructor & Destructor Documentation

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

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

Parameters

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

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

17.12.3 Member Function Documentation

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

Returns

Success of the construction.

17.12.3.2 DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix ()

Returns

The operator as a dense matrix.

17.12.4 Member Data Documentation

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

Definition at line 108 of file [mtk_interp_2d.h](#).

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

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

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

Definition at line 110 of file [mtk_interp_2d.h](#).

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

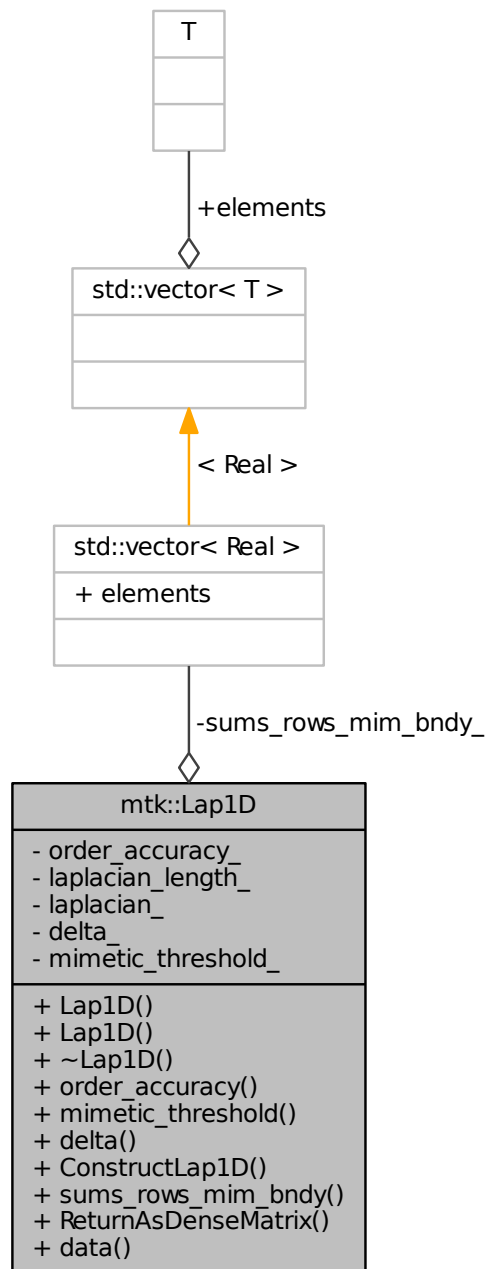
- [include/mtk_interp_2d.h](#)

17.13 mtk::Lap1D Class Reference

Implements a 1D mimetic Laplacian operator.

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


Collaboration diagram for mtk::Lap1D:



Public Member Functions

- [Lap1D](#) ()

- *Default constructor.*
- [Lap1D](#) (const [Lap1D](#) &lap)
- *Copy constructor.*
- [~Lap1D](#) ()
- *Destructor.*
- int [order_accuracy](#) () const
- *Order of accuracy of the operator.*
- [Real](#) [mimetic_threshold](#) () const
- *Mimetic threshold used in the CBS algorithm to construct this operator.*
- [Real](#) [delta](#) () const
- *Value of 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.*
- [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.*
- 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 112 of file [mtk_lap_1d.cc](#).

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

Parameters

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

17.13.2.3 mtk::Lap1D::~~Lap1D ()

Definition at line 118 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.

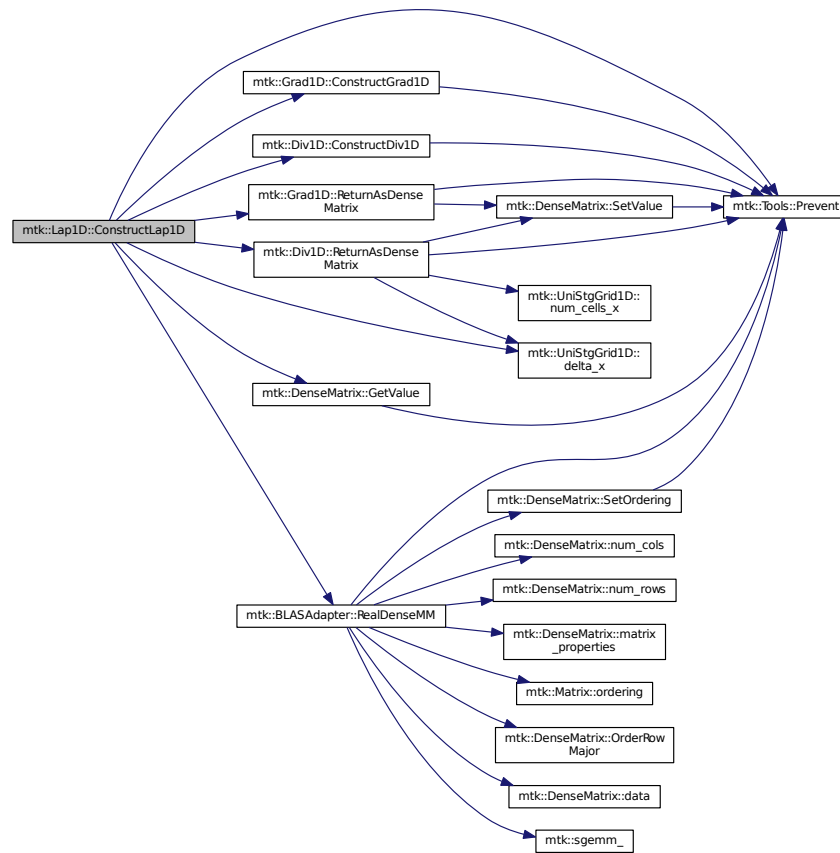
Warning

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

Here is the call graph for this function:



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

Returns

The operator as a dense array.

Definition at line 367 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 134 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:



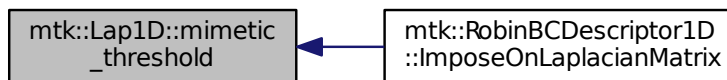
17.13.3.4 mtk::Real mtk::Lap1D::mimetic_threshold () const

Returns

Mimetic threshold used in the CBS algorithm to construct operator.

Definition at line 129 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:



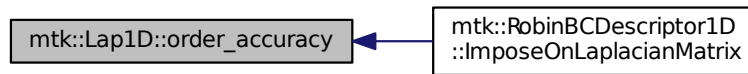
17.13.3.5 int mtk::Lap1D::order_accuracy () const

Returns

Order of accuracy of the operator.

Definition at line 124 of file [mtk_lap_1d.cc](#).

Here is the caller graph for this function:



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

Returns

The operator as a dense matrix.

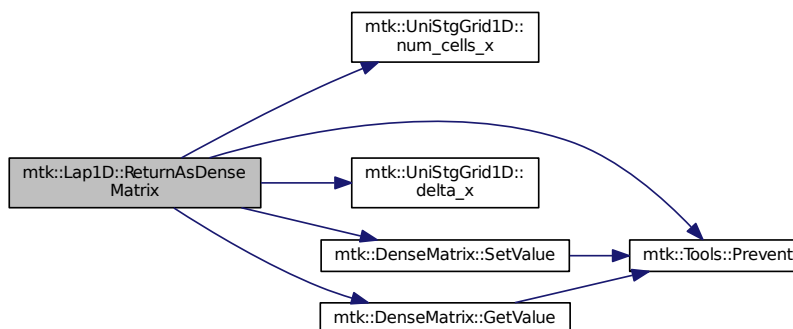
1. Extract mimetic coefficients from the west boundary.
2. Extract interior coefficients.
3. Extract mimetic coefficients from the west boundary to go east.

Note

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

Definition at line 297 of file [mtk_lap_1d.cc](#).

Here is the call graph for this function:



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

Returns

Collection of row-sums mimetic approximations at the boundary.

Definition at line 292 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 73 of file [mtk_lap_1d.cc](#).

17.13.5 Member Data Documentation

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

Definition at line 152 of file [mtk_lap_1d.h](#).

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

Definition at line 150 of file [mtk_lap_1d.h](#).

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

Definition at line 148 of file [mtk_lap_1d.h](#).

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

Definition at line 154 of file [mtk_lap_1d.h](#).

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

Definition at line 147 of file [mtk_lap_1d.h](#).

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

Definition at line 156 of file [mtk_lap_1d.h](#).

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

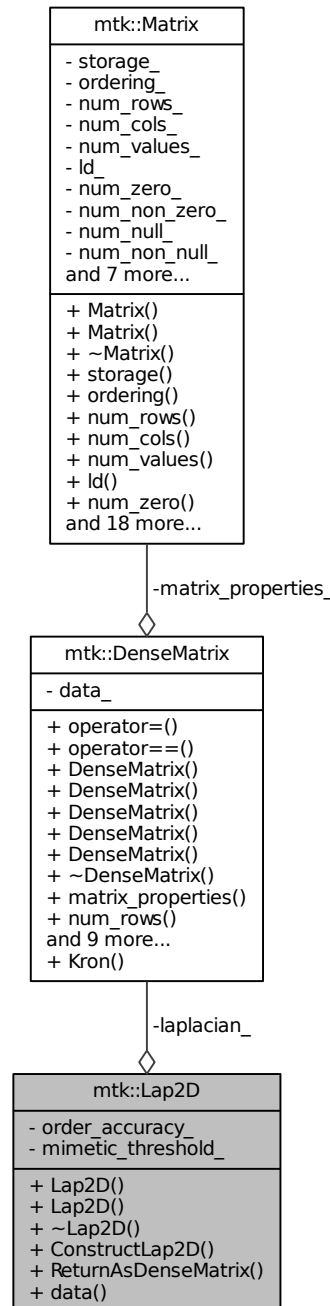
- [include/mtk_lap_1d.h](#)
- [src/mtk_lap_1d.cc](#)

17.14 mtk::Lap2D Class Reference

Implements a 2D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap2D:



Public Member Functions

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

Private Attributes

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

17.14.1 Detailed Description

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

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

17.14.2 Constructor & Destructor Documentation

17.14.2.1 mtk::Lap2D::Lap2D ()

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

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

Parameters

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

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

17.14.2.3 mtk::Lap2D::~~Lap2D ()

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

17.14.3 Member Function Documentation

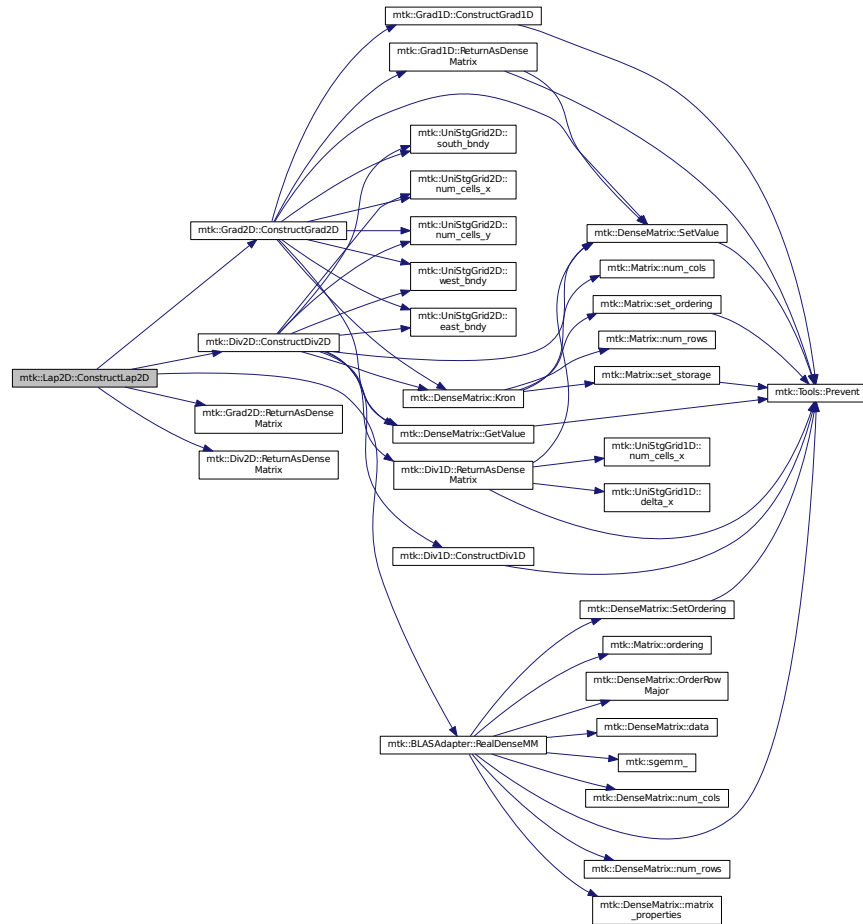
17.14.3.1 `bool mtk::Lap2D::ConstructLap2D (const UniStgGrid2D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

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

Here is the call graph for this function:



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

Returns

The operator as a dense array.

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

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

Returns

The operator as a dense matrix.

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

17.14.4 Member Data Documentation

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

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

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

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

17.14.4.3 **int** mtk::Lap2D::order_accuracy_ [private]

Definition at line 117 of file [mtk_lap_2d.h](#).

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

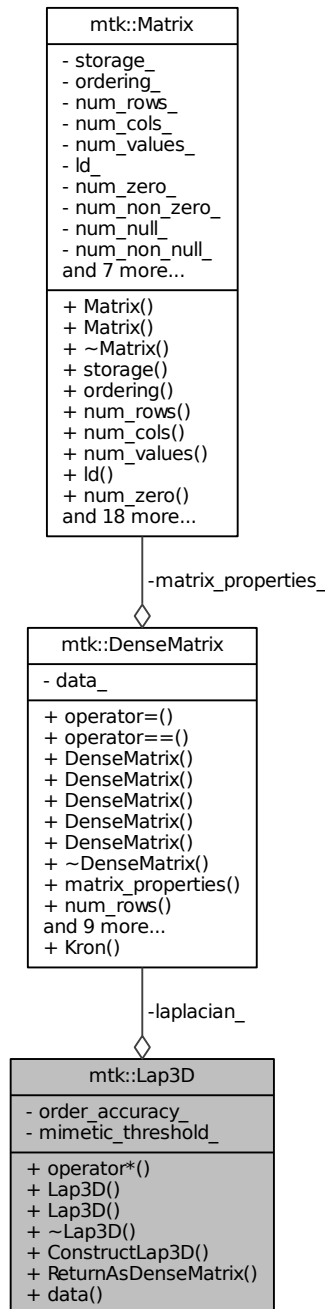
- [include/mtk_lap_2d.h](#)
- [src/mtk_lap_2d.cc](#)

17.15 mtk::Lap3D Class Reference

Implements a 3D mimetic Laplacian operator.

```
#include <mtk_lap_3d.h>
```

Collaboration diagram for mtk::Lap3D:



Public Member Functions

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

Operator application operator on a grid.

- [Lap3D](#) ()

Default constructor.

- [Lap3D](#) (const [Lap3D](#) &lap)

Copy constructor.

- [~Lap3D](#) ()

Destructor.

- bool [ConstructLap3D](#) (const [UniStgGrid3D](#) &grid, int order_accuracy=[kDefaultOrderAccuracy](#), [Real](#) mimetic_↵ threshold=[kDefaultMimeticThreshold](#))

Factory method implementing the CBS Algorithm to build operator.

- [DenseMatrix ReturnAsDenseMatrix](#) () const

Return the operator as a dense matrix.

- [Real * data](#) () const

Return the operator as a dense array.

Private Attributes

- [DenseMatrix laplacian_](#)

Actual operator.

- int [order_accuracy_](#)

Order of accuracy.

- [Real mimetic_threshold_](#)

Mimetic Threshold.

17.15.1 Detailed Description

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

Definition at line 76 of file [mtk_lap_3d.h](#).

17.15.2 Constructor & Destructor Documentation

17.15.2.1 mtk::Lap3D::Lap3D ()

Definition at line 76 of file [mtk_lap_3d.cc](#).

17.15.2.2 mtk::Lap3D::Lap3D (const Lap3D & lap)

Parameters

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

Definition at line 78 of file [mtk_lap_3d.cc](#).

17.15.2.3 mtk::Lap3D::~~Lap3D ()

Definition at line 82 of file [mtk_lap_3d.cc](#).

17.15.3 Member Function Documentation

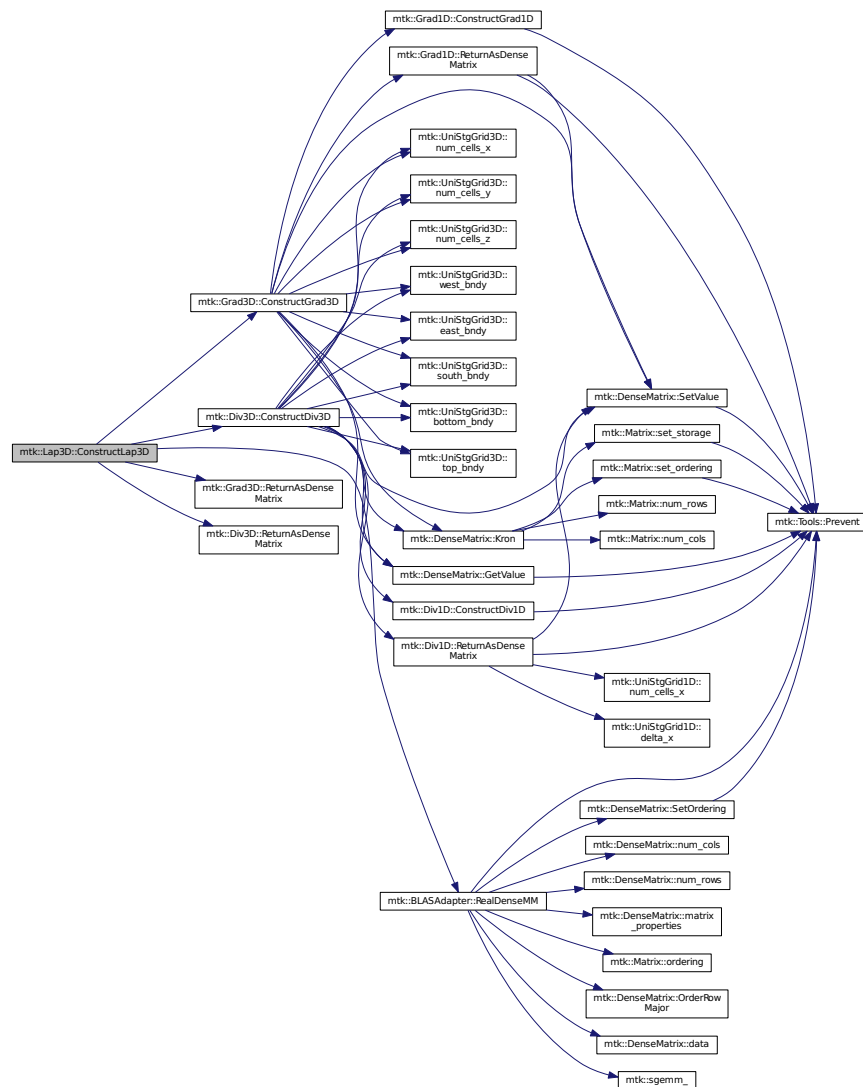
17.15.3.1 `bool mtk::Lap3D::ConstructLap3D (const UniStgGrid3D & grid, int order_accuracy = kDefaultOrderAccuracy, mtk::Real mimetic_threshold = kDefaultMimeticThreshold)`

Returns

Success of the construction.

Definition at line 84 of file `mtk_lap_3d.cc`.

Here is the call graph for this function:



17.15.3.2 `mtk::Real * mtk::Lap3D::data () const`

Returns

The operator as a dense array.

Definition at line 122 of file [mtk_lap_3d.cc](#).

17.15.3.3 `mtk::UniStgGrid3D mtk::Lap3D::operator* (const UniStgGrid3D & grid) const`

Definition at line 69 of file [mtk_lap_3d.cc](#).

17.15.3.4 `mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix () const`

Returns

The operator as a dense matrix.

Definition at line 117 of file [mtk_lap_3d.cc](#).

17.15.4 Member Data Documentation

17.15.4.1 `DenseMatrix mtk::Lap3D::laplacian_ [private]`

Definition at line 118 of file [mtk_lap_3d.h](#).

17.15.4.2 `Real mtk::Lap3D::mimetic_threshold_ [private]`

Definition at line 122 of file [mtk_lap_3d.h](#).

17.15.4.3 `int mtk::Lap3D::order_accuracy_ [private]`

Definition at line 120 of file [mtk_lap_3d.h](#).

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

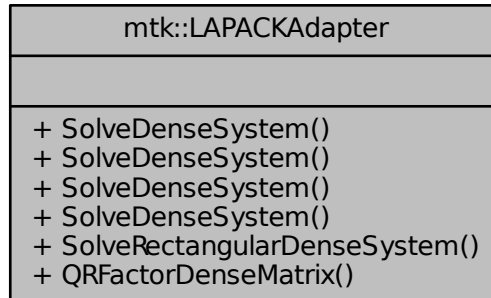
- [include/mtk_lap_3d.h](#)
- [src/mtk_lap_3d.cc](#)

17.16 mtk::LAPACKAdapter Class Reference

Adapter class for the LAPACK API.

```
#include <mtk_lapack_adapter.h>
```

Collaboration diagram for mtk::LAPACKAdapter:



Static Public Member Functions

- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::Real](#) *rhs)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::DenseMatrix](#) &rr)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid1D](#) &rhs)
Solves a dense system of linear equations.
- static int [SolveDenseSystem](#) ([mtk::DenseMatrix](#) &mm, [mtk::UniStgGrid2D](#) &rhs)
Solves a dense system of linear equations.
- static int [SolveRectangularDenseSystem](#) (const [mtk::DenseMatrix](#) &aa, [mtk::Real](#) *ob_, int ob_Id_)
Solves overdetermined or underdetermined real linear systems.
- static [mtk::DenseMatrix](#) [QRFactorDenseMatrix](#) ([DenseMatrix](#) &matrix)
Performs a QR factorization on a dense matrix.

17.16.1 Detailed Description

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

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

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

Definition at line 94 of file [mtk_lapack_adapter.h](#).

17.16.2 Member Function Documentation

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

Adapts the MTK to LAPACK's routine.

Parameters

| | | |
|----------------------|---------------------|---------------|
| <code>in, out</code> | <code>matrix</code> | Input matrix. |
|----------------------|---------------------|---------------|

Returns

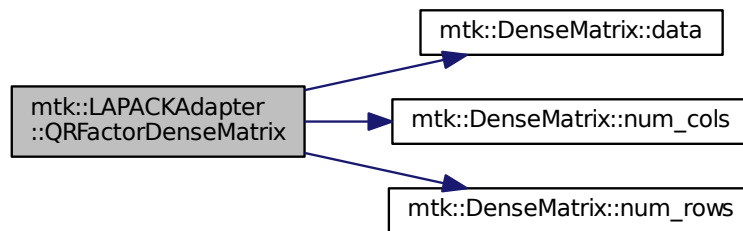
Matrix **Q**.

Exceptions

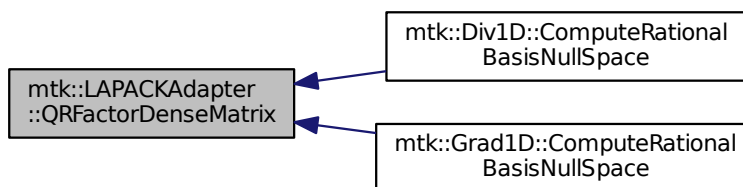
| |
|-----------------------------|
| <code>std::bad_alloc</code> |
|-----------------------------|

Definition at line 594 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

Parameters

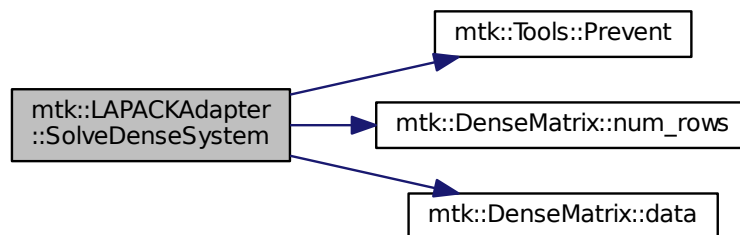
| | | |
|----|---------------|--------------------------------|
| in | <i>matrix</i> | Input matrix. |
| in | <i>rhs</i> | Input right-hand sides vector. |

Exceptions

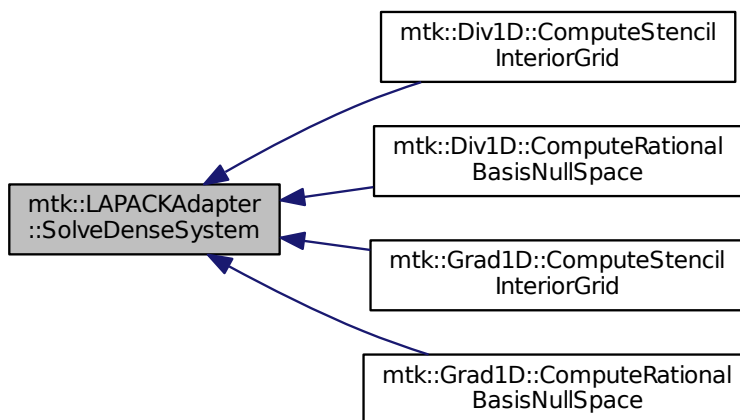
| | |
|-----------------------|--|
| <i>std::bad_alloc</i> | |
|-----------------------|--|

Definition at line 431 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

Parameters

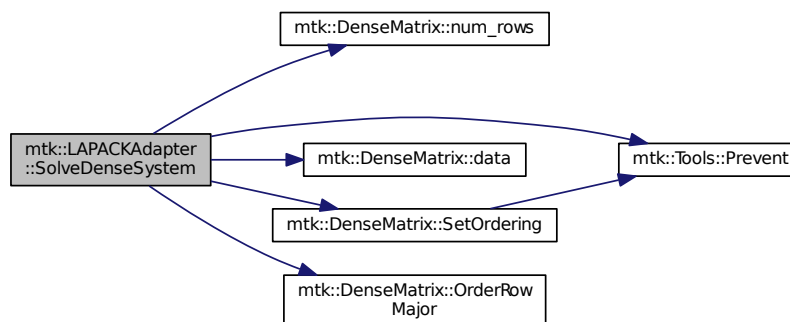
| | | |
|----|---------------|--------------------------------|
| in | <i>matrix</i> | Input matrix. |
| in | <i>rr</i> | Input right-hand sides matrix. |

Exceptions

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

Definition at line 466 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



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

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

Parameters

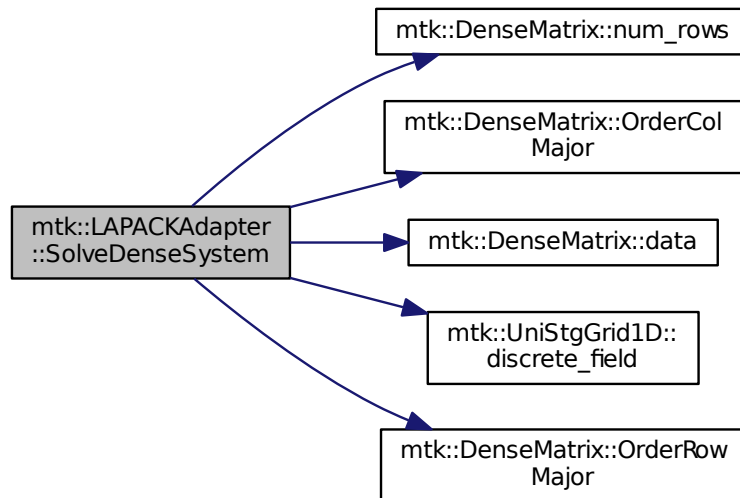
| | | |
|----|---------------|--------------------------------------------|
| in | <i>matrix</i> | Input matrix. |
| in | <i>rhs</i> | Input right-hand side from info on a grid. |

Exceptions

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

Definition at line 518 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



17.16.2.5 `int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & mm, mtk::UniStgGrid2D & rhs)`
`[static]`

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

Parameters

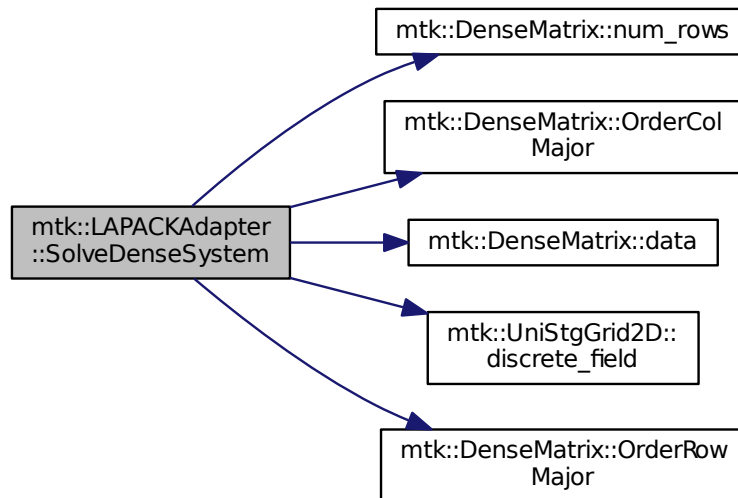
| | | |
|-----------------|---------------------|--------------------------------------------|
| <code>in</code> | <code>matrix</code> | Input matrix. |
| <code>in</code> | <code>rhs</code> | Input right-hand side from info on a grid. |

Exceptions

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

Definition at line 556 of file `mtk_lapack_adapter.cc`.

Here is the call graph for this function:



17.16.2.6 `int mtk::LAPACKAdapter::SolveRectangularDenseSystem (const mtk::DenseMatrix & aa, mtk::Real * ob_, int ob_id_) [static]`

Adapts the MTK to LAPACK's routine.

Parameters

| | | |
|----------------------|---------------------|---------------|
| <code>in, out</code> | <code>matrix</code> | Input matrix. |
|----------------------|---------------------|---------------|

Returns

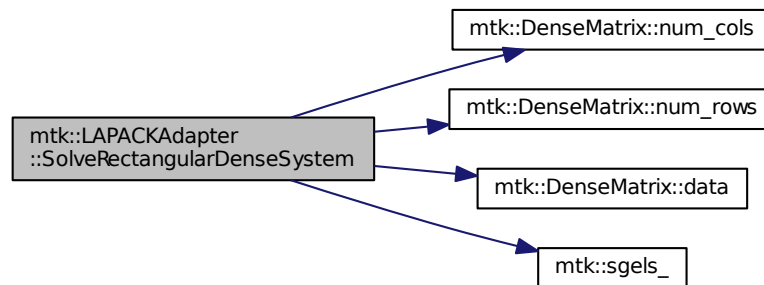
Success of the solution.

Exceptions

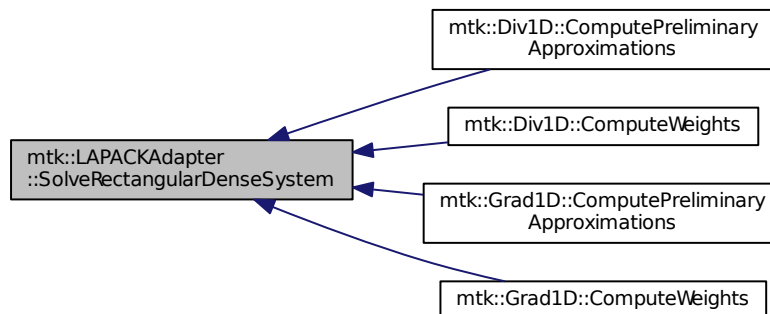
| | |
|-----------------------------|--|
| <code>std::bad_alloc</code> | |
|-----------------------------|--|

Definition at line 791 of file [mtk_lapack_adapter.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

17.17 mtk::Matrix Class Reference

Definition of the representation of a matrix in the MTK.

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

Collaboration diagram for mtk::Matrix:

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

Public Member Functions

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

- int [num_values](#) () const noexcept
Gets the number of values.
- int [ld](#) () const noexcept
Gets the matrix' leading dimension.
- int [num_zero](#) () const noexcept
Gets the number of zeros.
- int [num_non_zero](#) () const noexcept
Gets the number of non-zero values.
- int [num_null](#) () const noexcept
Gets the number of null values.
- int [num_non_null](#) () const noexcept
Gets the number of non-null values.
- int [kl](#) () const noexcept
Gets the number of lower diagonals.
- int [ku](#) () const noexcept
Gets the number of upper diagonals.
- int [bandwidth](#) () const noexcept
Gets the bandwidth.
- [Real abs_density](#) () const noexcept
Gets the absolute density.
- [Real rel_density](#) () const noexcept
Gets the relative density.
- [Real abs_sparsity](#) () const noexcept
Gets the Absolute sparsity.
- [Real rel_sparsity](#) () const noexcept
Gets the Relative sparsity.
- void [set_storage](#) (const [MatrixStorage](#) &tt) noexcept
Sets the storage type of the matrix.
- void [set_ordering](#) (const [MatrixOrdering](#) &oo) noexcept
Sets the ordering of the matrix.
- void [set_num_rows](#) (const int &num_rows) noexcept
Sets the number of rows of the matrix.
- void [set_num_cols](#) (const int &num_cols) noexcept
Sets the number of columns of the matrix.
- void [set_num_zero](#) (const int &in) noexcept
Sets the number of zero values of the matrix that matter.
- void [set_num_null](#) (const int &in) noexcept
Sets the number of zero values of the matrix that DO NOT matter.
- void [IncreaseNumZero](#) () noexcept
Increases the number of values that equal zero but with meaning.
- void [IncreaseNumNull](#) () noexcept
Increases the number of values that equal zero but with no meaning.

Private Attributes

- [MatrixStorage storage_](#)
What type of matrix is this?
- [MatrixOrdering ordering_](#)
What kind of ordering is it following?
- int [num_rows_](#)
Number of rows.
- int [num_cols_](#)
Number of columns.
- int [num_values_](#)
Number of total values in matrix.
- int [ld_](#)
Elements between successive rows when row-major.
- int [num_zero_](#)
Number of zeros.
- int [num_non_zero_](#)
Number of non-zero values.
- int [num_null_](#)
Number of null (insignificant) values.
- int [num_non_null_](#)
Number of null (significant) values.
- int [kl_](#)
Number of lower diagonals on a banded matrix.
- int [ku_](#)
Number of upper diagonals on a banded matrix.
- int [bandwidth_](#)
Bandwidth of the matrix.
- [Real abs_density_](#)
Absolute density of matrix.
- [Real rel_density_](#)
Relative density of matrix.
- [Real abs_sparsity_](#)
Absolute sparsity of matrix.
- [Real rel_sparsity_](#)
Relative sparsity of matrix.

17.17.1 Detailed Description

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

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

17.17.2 Constructor & Destructor Documentation

17.17.2.1 `mtk::Matrix::Matrix ()`

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

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

Parameters

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

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

17.17.2.3 `mtk::Matrix::~~Matrix ()` [noexcept]

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

17.17.3 Member Function Documentation

17.17.3.1 `Real mtk::Matrix::abs_density () const` [noexcept]

See also

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

Returns

Absolute density of the matrix.

17.17.3.2 `mtk::Real mtk::Matrix::abs_sparsity () const` [noexcept]

See also

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

Returns

Absolute sparsity of the matrix.

Definition at line 177 of file [mtk_matrix.cc](#).

17.17.3.3 `int mtk::Matrix::bandwidth () const` [noexcept]

Returns

Bandwidth of the matrix.

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

17.17.3.4 `void mtk::Matrix::IncreaseNumNull ()` [noexcept]

Todo Review the definition of sparse matrices properties.

Definition at line 275 of file [mtk_matrix.cc](#).

17.17.3.5 void mtk::Matrix::IncreaseNumZero () [noexcept]

Todo Review the definition of sparse matrices properties.

Definition at line 265 of file [mtk_matrix.cc](#).

17.17.3.6 int mtk::Matrix::kl () const [noexcept]

Returns

Number of lower diagonals.

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

17.17.3.7 int mtk::Matrix::ku () const [noexcept]

Returns

Number of upper diagonals.

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

17.17.3.8 int mtk::Matrix::ld () const [noexcept]

Leading dimension of the data array is the number of elements between successive rows (for row major storage) in memory. Most of the cases, the leading dimension is the same as the number of columns.

Returns

Leading dimension of the matrix.

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

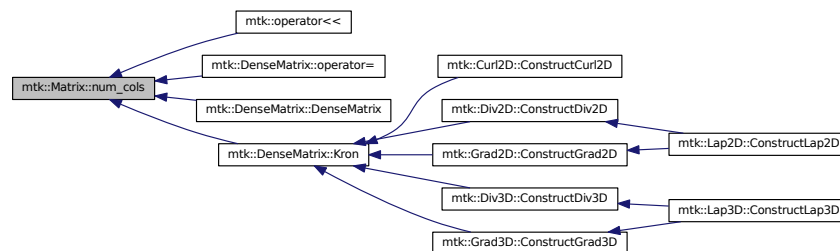
17.17.3.9 int mtk::Matrix::num_cols () const [noexcept]

Returns

Number of rows of the matrix.

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

Here is the caller graph for this function:



17.17.3.10 `int mtk::Matrix::num_non_null () const` [noexcept]

See also

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

Returns

Number of non-null values of the matrix.

Definition at line 152 of file `mtk_matrix.cc`.

17.17.3.11 `int mtk::Matrix::num_non_zero () const` [noexcept]

Returns

Number of non-zero values of the matrix.

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

17.17.3.12 `int mtk::Matrix::num_null () const` [noexcept]

See also

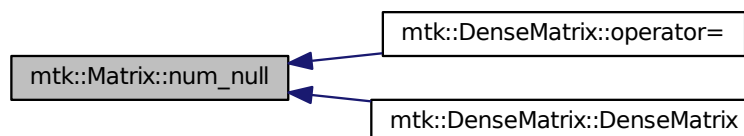
http://www.csrc.sdsu.edu/research_reports/CSR2013-01.pdf

Returns

Number of null values of the matrix.

Definition at line 147 of file `mtk_matrix.cc`.

Here is the caller graph for this function:



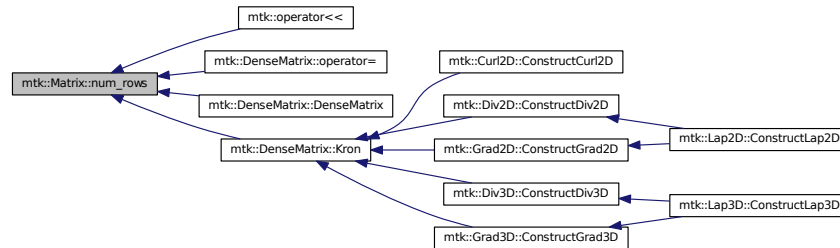
17.17.3.13 `int mtk::Matrix::num_rows () const` [noexcept]

Returns

Number of rows of the matrix.

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

Here is the caller graph for this function:



17.17.3.14 `int mtk::Matrix::num_values () const [noexcept]`

Returns

Number of values of the matrix.

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

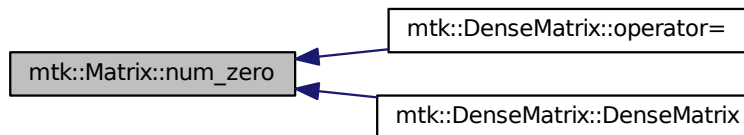
17.17.3.15 `int mtk::Matrix::num_zero () const [noexcept]`

Returns

Number of zeros of the matrix.

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

Here is the caller graph for this function:



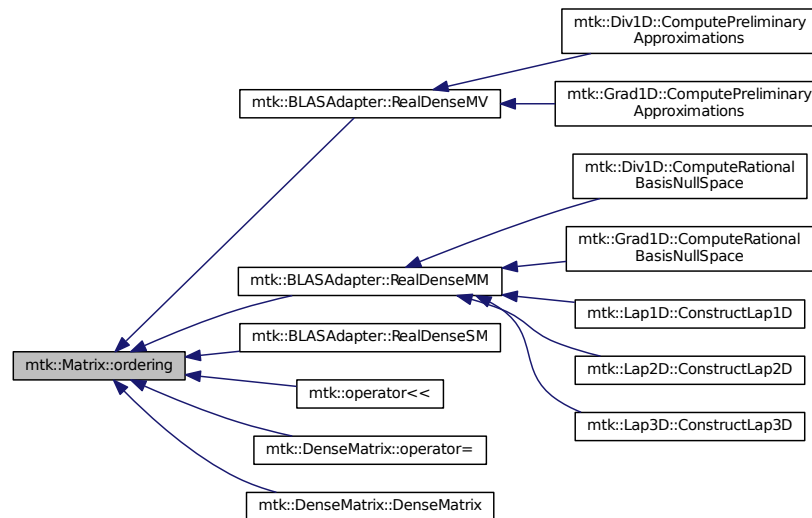
17.17.3.16 `mtk::MatrixOrdering mtk::Matrix::ordering () const [noexcept]`

Returns

Type of ordering of this matrix.

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

Here is the caller graph for this function:



17.17.3.17 `mtk::Real mtk::Matrix::rel_density () const` [noexcept]

See also

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

Returns

Relative density of the matrix.

Definition at line 172 of file [mtk_matrix.cc](#).

17.17.3.18 `mtk::Real mtk::Matrix::rel_sparsity () const` [noexcept]

See also

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

Returns

Relative sparsity of the matrix.

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

17.17.3.19 `void mtk::Matrix::set_num_cols (const int & num_cols)` [noexcept]

Parameters

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

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

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.20 `void mtk::Matrix::set_num_null (const int & in) [noexcept]`

Parameters

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

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

Definition at line 251 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.21 `void mtk::Matrix::set_num_rows (const int & num_rows) [noexcept]`

Parameters

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

Definition at line 213 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.22 `void mtk::Matrix::set_num_zero (const int & in) [noexcept]`

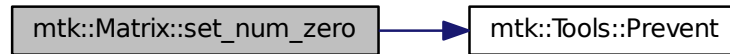
Parameters

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

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

Definition at line 237 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.23 `void mtk::Matrix::set_ordering (const MatrixOrdering & oo) [noexcept]`

See also

[MatrixOrdering](#)

Parameters

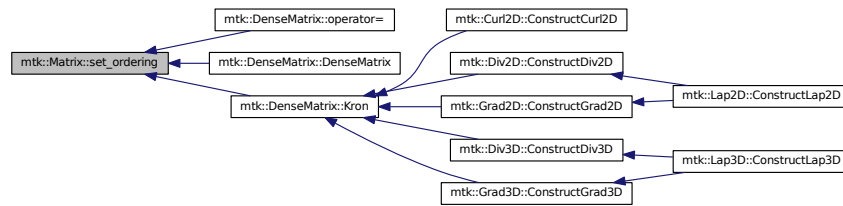
| | | |
|-----------------|-----------------|-------------------------|
| <code>in</code> | <code>oo</code> | Ordering of the matrix. |
|-----------------|-----------------|-------------------------|

Definition at line 199 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.17.3.24 `void mtk::Matrix::set_storage (const MatrixStorage & tt) [noexcept]`

See also

[MatrixStorage](#)

Parameters

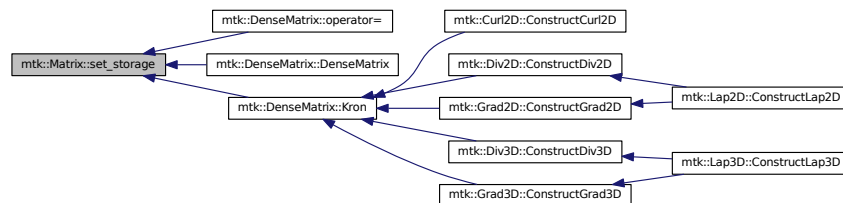
| | | |
|----|-----------|-----------------------------|
| in | <i>tt</i> | Type of the matrix storage. |
|----|-----------|-----------------------------|

Definition at line 187 of file [mtk_matrix.cc](#).

Here is the call graph for this function:



Here is the caller graph for this function:



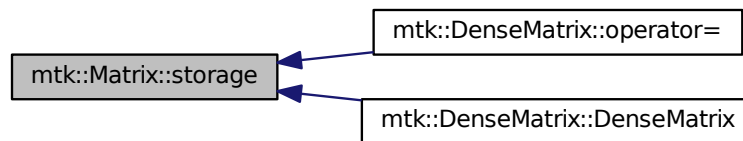
17.17.3.25 `mtk::MatrixStorage mtk::Matrix::storage () const [noexcept]`

Returns

Type of storage of this matrix.

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

Here is the caller graph for this function:

**17.17.4 Member Data Documentation****17.17.4.1 Real mtk::Matrix::abs_density_ [private]**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

17.17.4.17 **MatrixStorage** `mtk::Matrix::storage_ [private]`

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

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

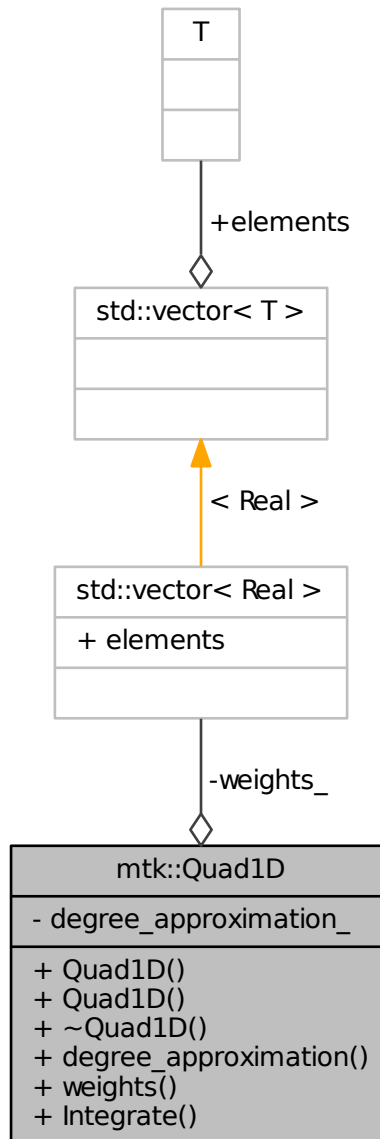
- [include/mtk_matrix.h](#)
- [src/mtk_matrix.cc](#)

17.18 mtk::Quad1D Class Reference

Implements a 1D mimetic quadrature.

```
#include <mtk_quad_1d.h>
```

Collaboration diagram for mtk::Quad1D:



Public Member Functions

- [Quad1D](#) ()
Default constructor.
- [Quad1D](#) (const [Quad1D](#) &quad)
Copy constructor.
- [~Quad1D](#) ()
Destructor.
- int [degree_approximation](#) () const
Get the degree of interpolating polynomial per sub-interval of domain.
- [Real](#) * [weights](#) () const
Return collection of weights.
- [Real](#) [Integrate](#) ([Real](#)(*Integrand)([Real](#) xx), [UniStgGrid1D](#) grid) const
Mimetic integration routine.

Private Attributes

- int [degree_approximation_](#)
Degree of the interpolating polynomial.
- std::vector< [Real](#) > [weights_](#)
Collection of weights.

Friends

- std::ostream & [operator<<](#) (std::ostream &stream, [Quad1D](#) &in)
Output stream operator for printing.

17.18.1 Detailed Description

This class implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

Definition at line 81 of file [mtk_quad_1d.h](#).

17.18.2 Constructor & Destructor Documentation

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

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

Parameters

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

17.18.2.3 mtk::Quad1D::~~Quad1D ()

17.18.3 Member Function Documentation

17.18.3.1 int mtk::Quad1D::degree_approximation () const

Returns

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

17.18.3.2 Real mtk::Quad1D::Integrate (Real(*) (Real xx) *Integrand*, UniStgGrid1D *grid*) const

Parameters

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

Returns

Result of the integration.

17.18.3.3 Real* mtk::Quad1D::weights () const

Returns

Collection of weights.

17.18.4 Friends And Related Function Documentation

17.18.4.1 std::ostream& operator<< (std::ostream & *stream*, Quad1D & *in*) [friend]

17.18.5 Member Data Documentation

17.18.5.1 int mtk::Quad1D::degree_approximation_ [private]

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

17.18.5.2 std::vector<Real> mtk::Quad1D::weights_ [private]

Definition at line 126 of file [mtk_quad_1d.h](#).

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

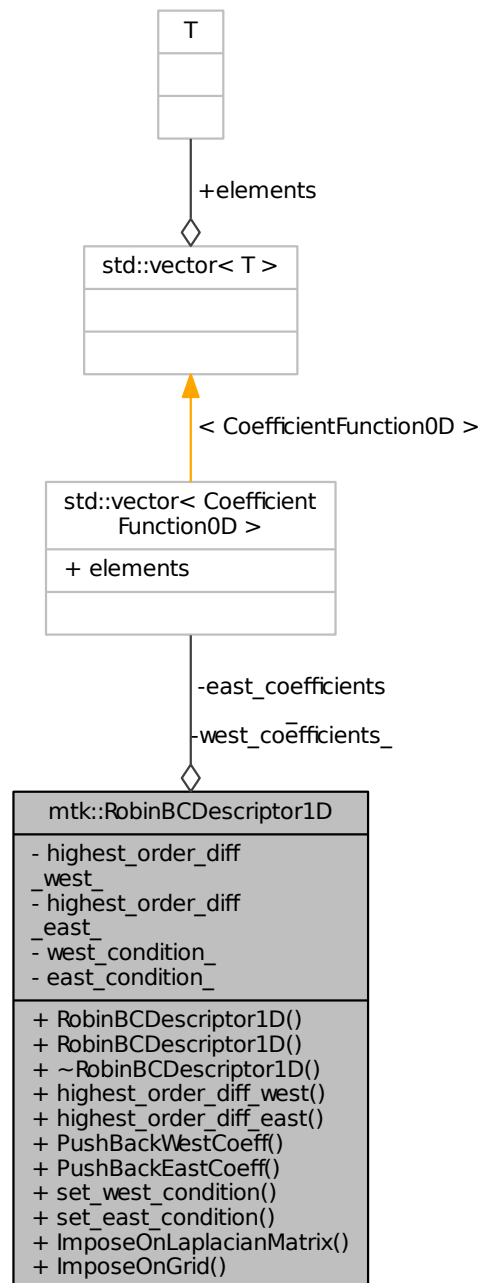
- [include/mtk_quad_1d.h](#)

17.19 mtk::RobinBCDescriptor1D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_1d.h>
```

Collaboration diagram for `mtk::RobinBCDescriptor1D`:



Public Member Functions

- [RobinBCDescriptor1D \(\)](#)

Default constructor.

- [RobinBCDescriptor1D](#) (const [RobinBCDescriptor1D](#) &desc)

Copy constructor.

- [~RobinBCDescriptor1D](#) () noexcept

Destructor.

- int [highest_order_diff_west](#) () const noexcept

Getter for the highest order of differentiation in the west boundary.

- int [highest_order_diff_east](#) () const noexcept

Getter for the highest order of differentiation in the east boundary.

- void [PushBackWestCoeff](#) ([CoefficientFunction0D](#) cw)

Push back coefficient function at west of lowest order diff. available.

- void [PushBackEastCoeff](#) ([CoefficientFunction0D](#) ce)

Push back coefficient function at east of lowest order diff. available.

- void [set_west_condition](#) ([Real](#)(*west_condition)(const [Real](#) &tt)) noexcept

Set boundary condition at west.

- void [set_east_condition](#) ([Real](#)(*east_condition)(const [Real](#) &tt)) noexcept

Set boundary condition at east.

- bool [ImposeOnLaplacianMatrix](#) (const [Lap1D](#) &lap, [DenseMatrix](#) &matrix, const [Real](#) &time=[mtk::kZero](#)) const

Imposes the condition on the operator represented as matrix.

- void [ImposeOnGrid](#) ([UniStgGrid1D](#) &grid, const [Real](#) &time=[mtk::kZero](#)) const

Imposes the condition on the grid.

Private Attributes

- int [highest_order_diff_west_](#)

Highest order of differentiation for west.

- int [highest_order_diff_east_](#)

Highest order of differentiation for east.

- std::vector

< [CoefficientFunction0D](#) > [west_coefficients_](#)

Coeffs. west.

- std::vector

< [CoefficientFunction0D](#) > [east_coefficients_](#)

Coeffs. east.

- [Real](#)(* [west_condition_](#))(const [Real](#) &tt)

Condition for west.

- [Real](#)(* [east_condition_](#))(const [Real](#) &tt)

Condition for east.

17.19.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) = \beta_b(b, t).$$

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 155 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.2 Constructor & Destructor Documentation

17.19.2.1 `mtk::RobinBCDescriptor1D::RobinBCDescriptor1D ()`

Definition at line 93 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.19.2.2 `mtk::RobinBCDescriptor1D::RobinBCDescriptor1D (const RobinBCDescriptor1D & desc)`

Parameters

| | | |
|-----------------|-------------------|----------------------|
| <code>in</code> | <code>desc</code> | Given 1D descriptor. |
|-----------------|-------------------|----------------------|

Definition at line 99 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.19.2.3 `mtk::RobinBCDescriptor1D::~~RobinBCDescriptor1D ()` `[noexcept]`

Definition at line 106 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.19.3 Member Function Documentation

17.19.3.1 `int mtk::RobinBCDescriptor1D::highest_order_diff_east () const` `[noexcept]`

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 113 of file [mtk_robin_bc_descriptor_1d.cc](#).

17.19.3.2 `int mtk::RobinBCDescriptor1D::highest_order_diff_west () const` `[noexcept]`

Returns

Integer highest order of differentiation in the west boundary.

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

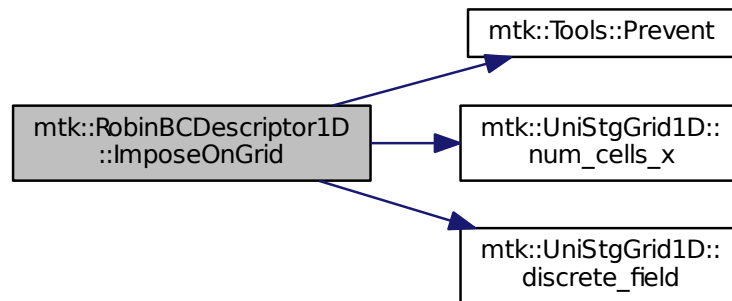
17.19.3.3 void mtk::RobinBCDescriptor1D::ImposeOnGrid (UniStgGrid1D & *grid*, const Real & *time* = mtk::kZero) const

Parameters

| | | |
|---------|-------------|--------------------------------------------------------|
| in, out | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

Definition at line 246 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



17.19.3.4 bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix (const Lap1D & *lap*, mtk::DenseMatrix & *matrix*, const Real & *time* = mtk::kZero) const

Parameters

| | | |
|---------|---------------|------------------------------------------|
| in | <i>lap</i> | Operator in the 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.

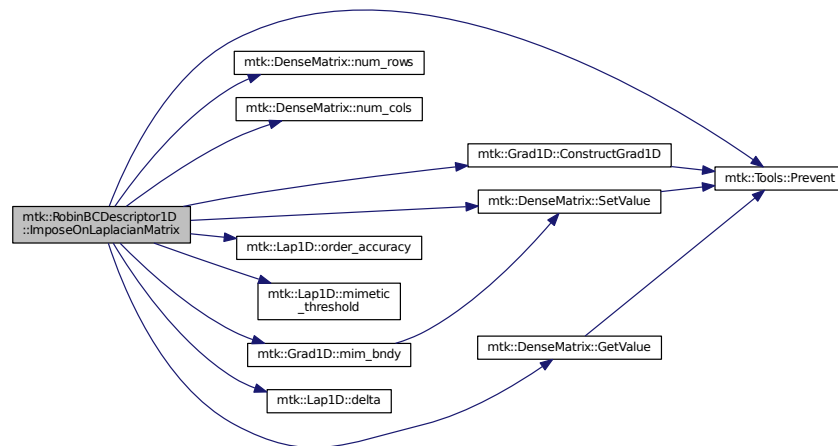
2.4.1. Get gradient coefficient and scale it.

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 166 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



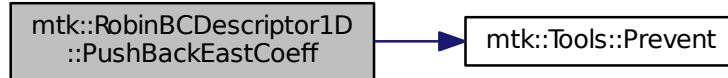
17.19.3.5 `void mtk::RobinBCDescriptor1D::PushBackEastCoeff (mtk::CoefficientFunction0D ce)`

Parameters

| | | |
|-----------|-----------|----------------------------------------------------|
| <i>in</i> | <i>ce</i> | Function $c_e(x, y) : \Omega \mapsto \mathbb{R}$. |
|-----------|-----------|----------------------------------------------------|

Definition at line 132 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



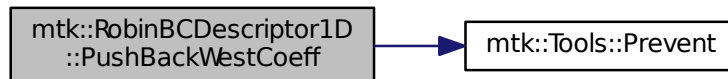
17.19.3.6 void mtk::RobinBCDescriptor1D::PushBackWestCoeff (mtk::CoefficientFunction0D *cw*)

Parameters

| | | |
|-----------|-----------|----------------------------------------------------|
| <i>in</i> | <i>cw</i> | Function $c_w(x, y) : \Omega \mapsto \mathbb{R}$. |
|-----------|-----------|----------------------------------------------------|

Definition at line 118 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



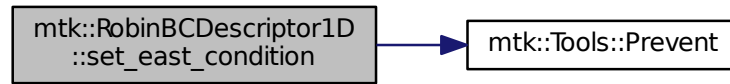
17.19.3.7 void mtk::RobinBCDescriptor1D::set_east_condition (Real(*) (const Real &tt) *east_condition*) [noexcept]

Parameters

| | | |
|-----------|-----------------------|-----------------------------------------------|
| <i>in</i> | <i>east_condition</i> | $\beta_e(y, t) : \Omega \mapsto \mathbb{R}$. |
|-----------|-----------------------|-----------------------------------------------|

Definition at line 156 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



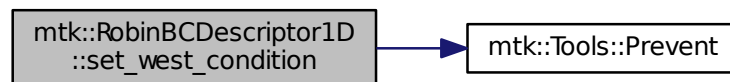
17.19.3.8 void mtk::RobinBCDescriptor1D::set_west_condition (Real(*) (const Real &tt) *west_condition*) [noexcept]

Parameters

| | | |
|----|-----------------------|----------------------------------------------|
| in | <i>west_condition</i> | $\beta_w(y, t) : \Omega \mapsto \mathbb{R}.$ |
|----|-----------------------|----------------------------------------------|

Definition at line 146 of file [mtk_robin_bc_descriptor_1d.cc](#).

Here is the call graph for this function:



17.19.4 Member Data Documentation

17.19.4.1 std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::east_coefficients_ [private]

Definition at line 237 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.4.2 Real(* mtk::RobinBCDescriptor1D::east_condition_)(const Real &tt) [private]

Definition at line 240 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.4.3 int mtk::RobinBCDescriptor1D::highest_order_diff_east_ [private]

Definition at line 234 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.4.4 int mtk::RobinBCDescriptor1D::highest_order_diff_west_ [private]

Definition at line 233 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.4.5 `std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::west_coefficients_` [private]

Definition at line 236 of file [mtk_robin_bc_descriptor_1d.h](#).

17.19.4.6 `Real(* mtk::RobinBCDescriptor1D::west_condition_)(const Real &tt)` [private]

Definition at line 239 of file [mtk_robin_bc_descriptor_1d.h](#).

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

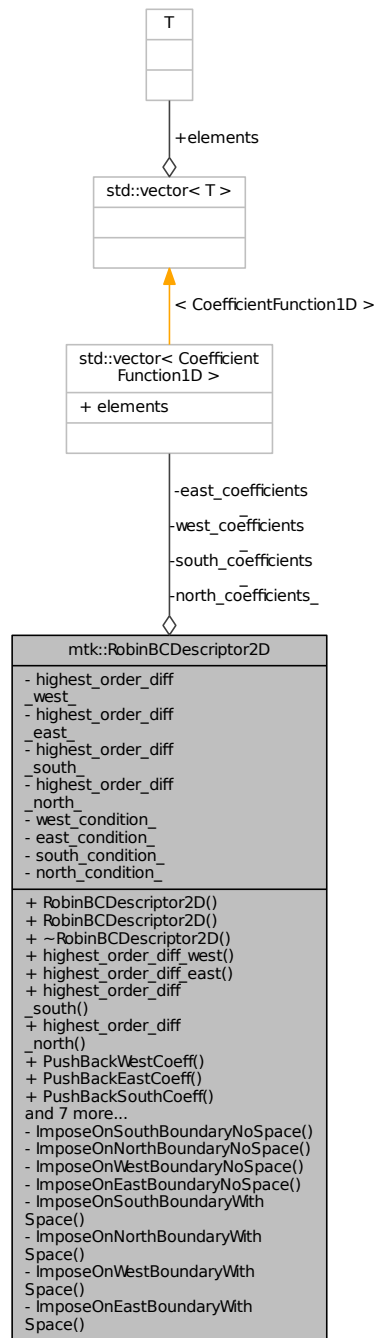
- [include/mtk_robin_bc_descriptor_1d.h](#)
- [src/mtk_robin_bc_descriptor_1d.cc](#)

17.20 mtk::RobinBCDescriptor2D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_2d.h>
```

Collaboration diagram for `mtk::RobinBCDescriptor2D`:



Public Member Functions

- [RobinBCDescriptor2D \(\)](#)

Default constructor.

- [RobinBCDescriptor2D](#) (const [RobinBCDescriptor2D](#) &desc)

Copy constructor.

- [~RobinBCDescriptor2D](#) () noexcept

Destructor.

- int [highest_order_diff_west](#) () const noexcept

Getter for the highest order of differentiation in the west boundary.

- int [highest_order_diff_east](#) () const noexcept

Getter for the highest order of differentiation in the east boundary.

- int [highest_order_diff_south](#) () const noexcept

Getter for the highest order of differentiation in the south boundary.

- int [highest_order_diff_north](#) () const noexcept

Getter for the highest order of differentiation in the north boundary.

- void [PushBackWestCoeff](#) ([CoefficientFunction1D](#) cw)

Push back coefficient function at west of lowest order diff. available.

- void [PushBackEastCoeff](#) ([CoefficientFunction1D](#) ce)

Push back coefficient function at east of lowest order diff. available.

- void [PushBackSouthCoeff](#) ([CoefficientFunction1D](#) cs)

Push back coefficient function south of lowest order diff. available.

- void [PushBackNorthCoeff](#) ([CoefficientFunction1D](#) cn)

Push back coefficient function north of lowest order diff. available.

- void [set_west_condition](#) ([Real](#)(*west_condition)(const [Real](#) &yy, const [Real](#) &tt)) noexcept

Set boundary condition at west.

- void [set_east_condition](#) ([Real](#)(*east_condition)(const [Real](#) &yy, const [Real](#) &tt)) noexcept

Set boundary condition at east.

- void [set_south_condition](#) ([Real](#)(*south_condition)(const [Real](#) &xx, const [Real](#) &tt)) noexcept

Set boundary condition at south.

- void [set_north_condition](#) ([Real](#)(*north_condition)(const [Real](#) &xx, const [Real](#) &tt)) noexcept

Set boundary condition at north.

- bool [ImposeOnLaplacianMatrix](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

Imposes the condition on the operator represented as matrix.

- void [ImposeOnGrid](#) ([UniStgGrid2D](#) &grid, const [Real](#) &time=kZero) const

Imposes the condition on the grid.

Private Member Functions

- bool [ImposeOnSouthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

Imposes the condition on the south boundary.

- bool [ImposeOnNorthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

Imposes the condition on the north boundary.

- bool [ImposeOnWestBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

Imposes the condition on the west boundary.

- bool [ImposeOnEastBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=kZero) const

Imposes the condition on the east boundary.

- bool [ImposeOnSouthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the south boundary.

- bool [ImposeOnNorthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the north boundary.

- bool [ImposeOnWestBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the west boundary.

- bool [ImposeOnEastBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the east boundary.

Private Attributes

- int [highest_order_diff_west_](#)

Highest order of differentiation west.

- int [highest_order_diff_east_](#)

Highest order of differentiation east.

- int [highest_order_diff_south_](#)

Highest order differentiation for south.

- int [highest_order_diff_north_](#)

Highest order differentiation for north.

- std::vector
< [CoefficientFunction1D](#) > [west_coefficients_](#)
Coeffs. west.

- std::vector
< [CoefficientFunction1D](#) > [east_coefficients_](#)
Coeffs. east.

- std::vector
< [CoefficientFunction1D](#) > [south_coefficients_](#)
Coeffs. south.

- std::vector
< [CoefficientFunction1D](#) > [north_coefficients_](#)
Coeffs. north.

- [Real](#)(* [west_condition_](#))(const [Real](#) &xx, const [Real](#) &tt)
Condition west.

- [Real](#)(* [east_condition_](#))(const [Real](#) &xx, const [Real](#) &tt)
Condition east.

- [Real](#)(* [south_condition_](#))(const [Real](#) &yy, const [Real](#) &tt)
Cond. south.

- [Real](#)(* [north_condition_](#))(const [Real](#) &yy, const [Real](#) &tt)
Cond. north.

17.20.1 Detailed Description

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

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 132 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.2 Constructor & Destructor Documentation

17.20.2.1 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D ()

Definition at line 84 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.2.2 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D (const RobinBCDescriptor2D & desc)

Parameters

| | | |
|-----------|-------------|----------------------|
| <i>in</i> | <i>desc</i> | Given 2D descriptor. |
|-----------|-------------|----------------------|

Definition at line 94 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.2.3 mtk::RobinBCDescriptor2D::~~RobinBCDescriptor2D () [noexcept]

Definition at line 105 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.3 Member Function Documentation

17.20.3.1 int mtk::RobinBCDescriptor2D::highest_order_diff_east () const [noexcept]

Returns

Integer highest order of differentiation in the east boundary.

Definition at line 112 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.3.2 `int mtk::RobinBCDescriptor2D::highest_order_diff_north () const` `[noexcept]`

Returns

Integer highest order of differentiation in the north boundary.

Definition at line 122 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.3.3 `int mtk::RobinBCDescriptor2D::highest_order_diff_south () const` `[noexcept]`

Returns

Integer highest order of differentiation in the south boundary.

Definition at line 117 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.3.4 `int mtk::RobinBCDescriptor2D::highest_order_diff_west () const` `[noexcept]`

Returns

Integer highest order of differentiation in the west boundary.

Definition at line 107 of file [mtk_robin_bc_descriptor_2d.cc](#).

17.20.3.5 `bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const` `[private]`

Parameters

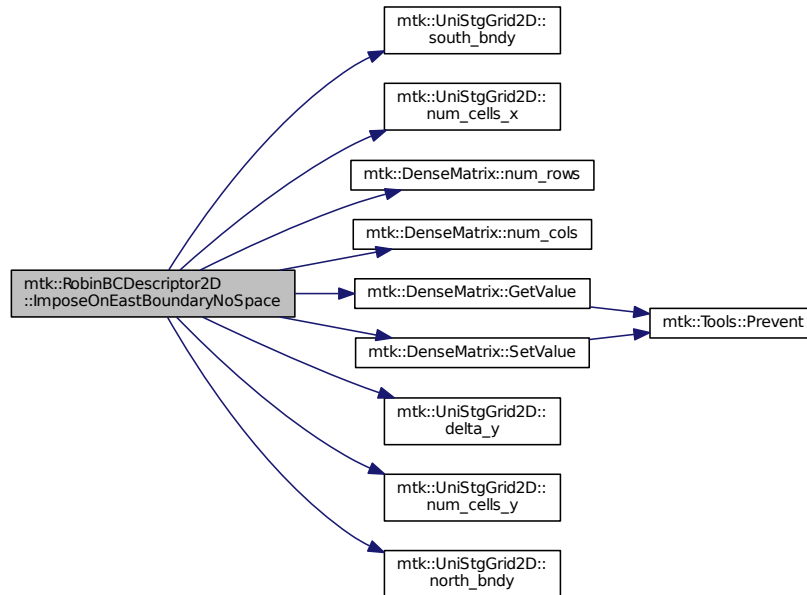
| | | |
|----------------|---------------|--------------------------------------------------------|
| <i>in</i> | <i>lap</i> | Laplacian operator on the matrix. |
| <i>in</i> | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| <i>in, out</i> | <i>matrix</i> | Input matrix with the Laplacian operator. |
| <i>in</i> | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.

2. Impose the Neumann condition.

Definition at line 495 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.6 `bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

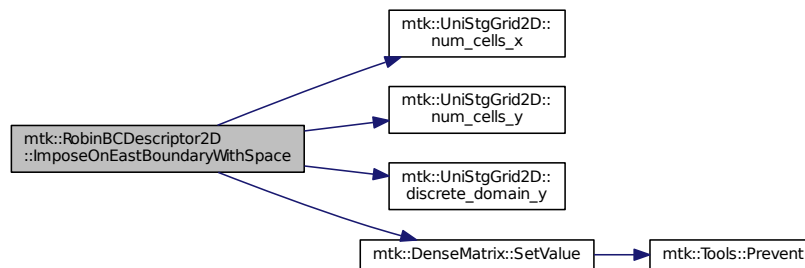
Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 564 of file `mtk_robin_bc_descriptor_2d.cc`.

Here is the call graph for this function:



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

Parameters

| | | |
|---------|-------------|--------------------------------------------------------|
| in, out | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose assuming an scalar grid.

1.1. Impose south condition.

1.1.1. Impose south-west corner.

1.1.2. Impose south border.

1.1.3. Impose south-east corner.

1.2. Impose north condition.

1.2.1. Impose north-west corner.

1.2.2. Impose north border.

1.2.3. Impose north-east corner.

1.3. Impose west condition.

1.3.1. Impose south-west corner.

Note

As per discussion with Otilio, we will take the **arithmetic mean** of the values of the BCs at the corners.

1.3.2. Impose west border.

1.3.3. Impose north-west corner.

1.4. Impose east condition.

1.4.1. Impose south-east corner.

1.4.2. Impose east border.

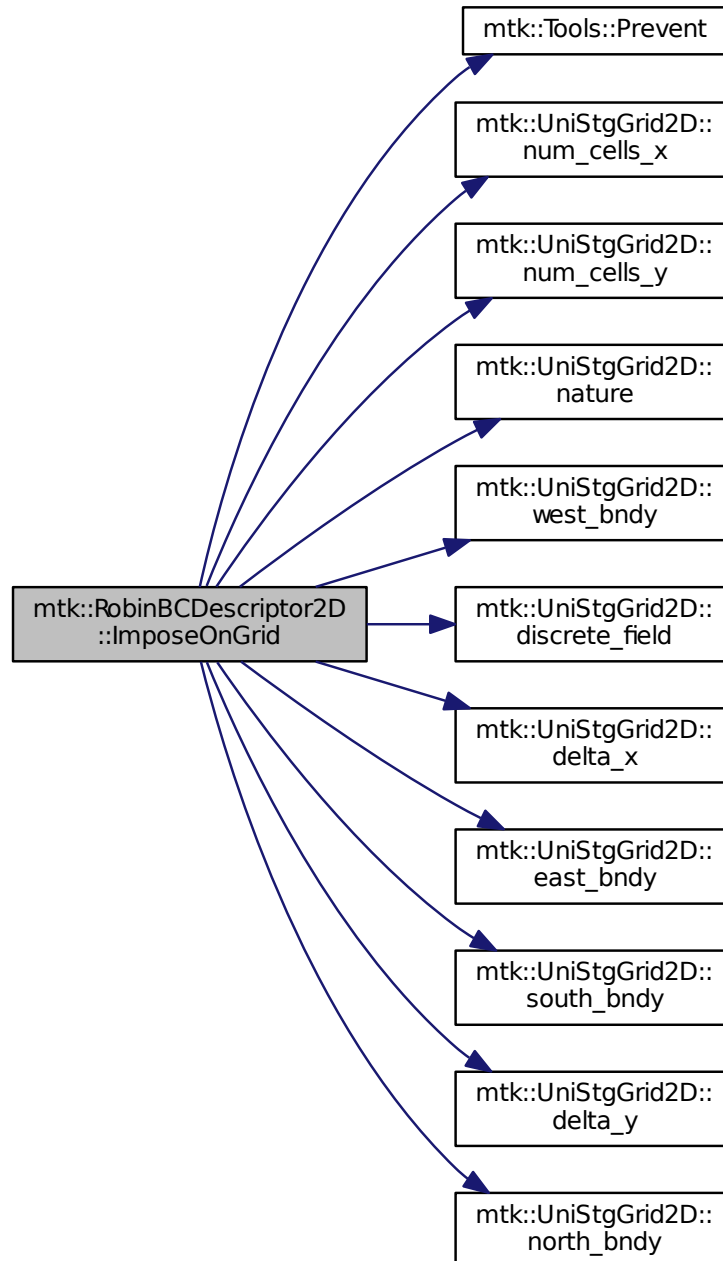
1.4.3. Impose north-east corner.

1. Impose assuming a vector grid.

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

Definition at line 674 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.8 `bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const`

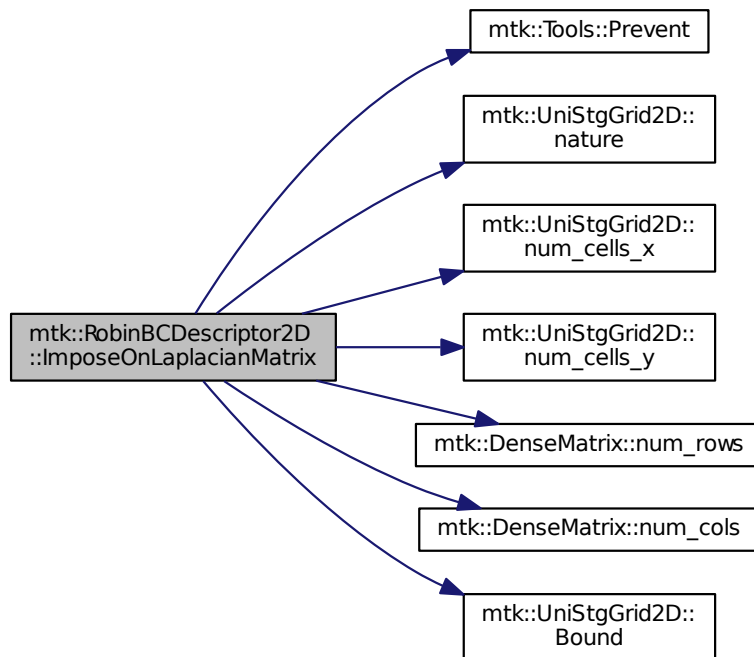
Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

If we have not bound anything to the grid, then we have to generate our collection of spatial coordinates, as we evaluate the coefficients.

Definition at line 591 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.9 `bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

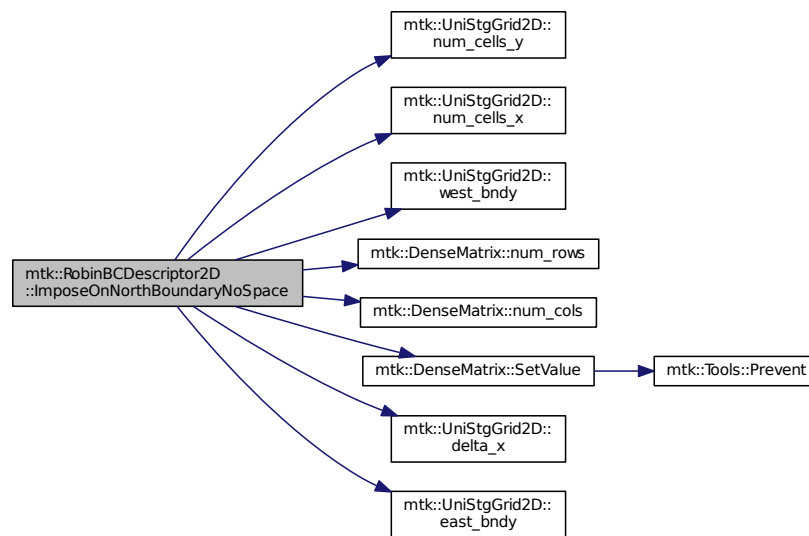
Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 312 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.10 `bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const` [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose Dirichlet condition.

For each entry on the diagonal:

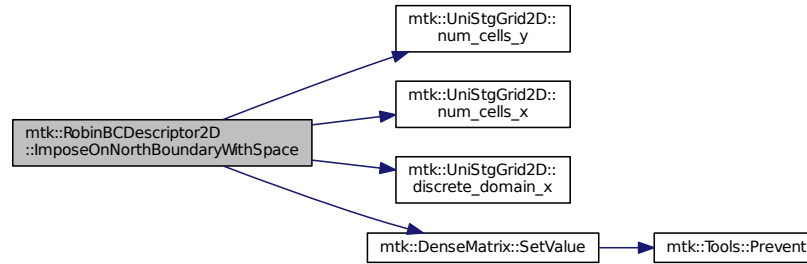
Evaluate next set spatial coordinates to evaluate the coefficient.

Evaluate and assign the Dirichlet coefficient.

1. Impose the Neumann condition.

Definition at line 372 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.11 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

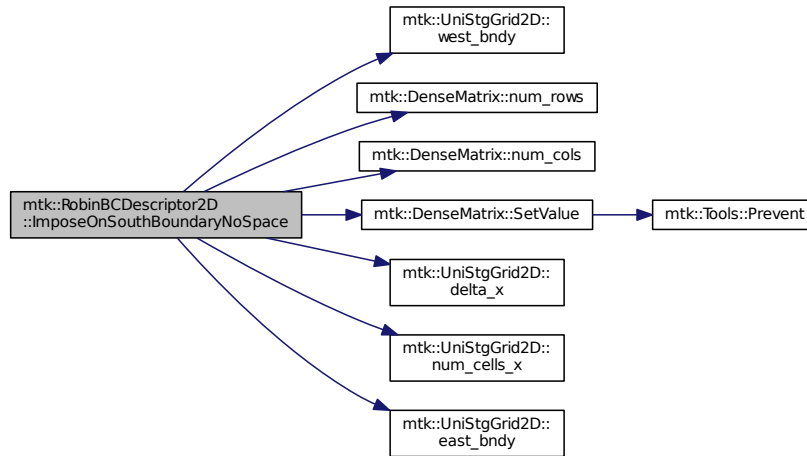
| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

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

Definition at line 229 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.12 `bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

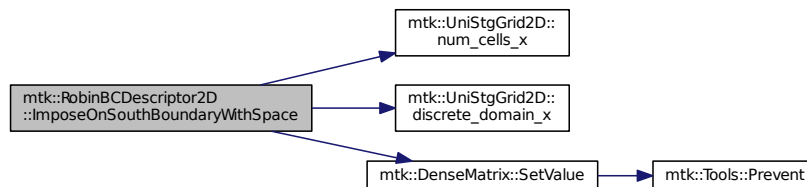
1. Impose the Dirichlet condition first.

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

1. Impose the Neumann condition.

Definition at line 284 of file `mtk_robin_bc_descriptor_2d.cc`.

Here is the call graph for this function:



17.20.3.13 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.

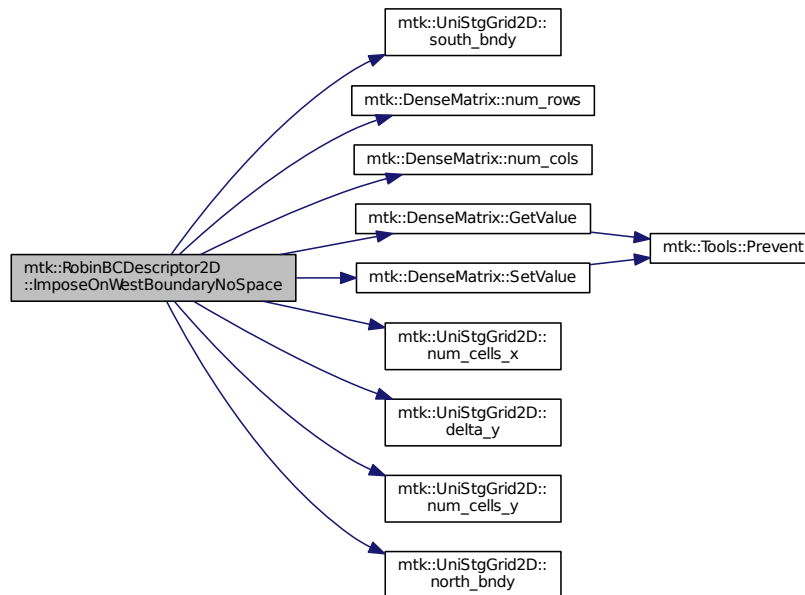
Note

As it can be seen, we must adopt a convention about how to treat the corners. Based on a reasoning with Otilio, we will take the **harmonic mean**.

1. Impose the Neumann condition.

Definition at line 399 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.3.14 `bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace (const Lap2D & lap, const UniStgGrid2D & grid, mtk::DenseMatrix & matrix, const Real & time = kZero) const [private]`

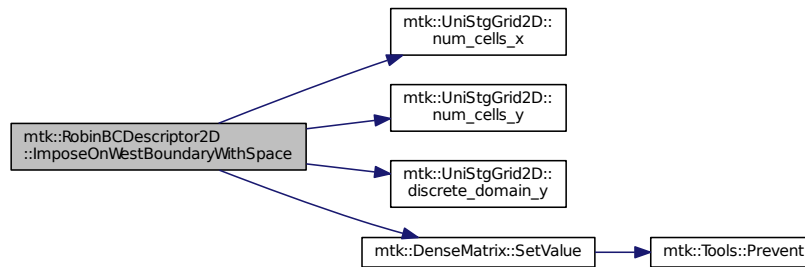
Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

1. Impose the Dirichlet condition first.
2. Impose the Neumann condition.

Definition at line 468 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



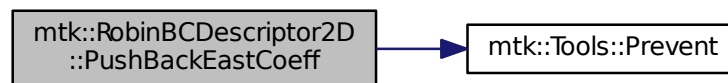
17.20.3.15 void mtk::RobinBCDescriptor2D::PushBackEastCoeff (mtk::CoefficientFunction1D *ce*)

Parameters

| | | |
|----|-----------|----------------------------------------------------------------------------|
| in | <i>cw</i> | Coeff. $c_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|----|-----------|----------------------------------------------------------------------------|

Definition at line 141 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



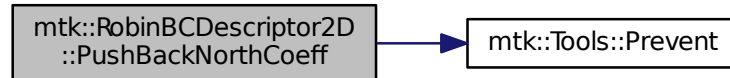
17.20.3.16 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff (mtk::CoefficientFunction1D *cn*)

Parameters

| | | |
|----|----|----------------------------------------------------------------------------|
| in | cw | Coeff. $c_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|----|----|----------------------------------------------------------------------------|

Definition at line 169 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



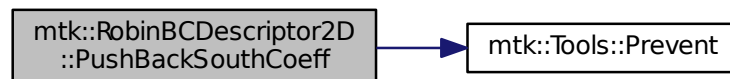
17.20.3.17 void `mtk::RobinBCDescriptor2D::PushBackSouthCoeff` (`mtk::CoefficientFunction1D cs`)

Parameters

| | | |
|----|----|----------------------------------------------------------------------------|
| in | cw | Coeff. $c_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|----|----|----------------------------------------------------------------------------|

Definition at line 155 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



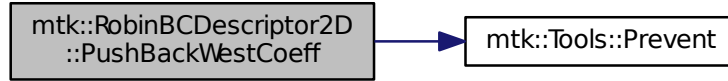
17.20.3.18 void `mtk::RobinBCDescriptor2D::PushBackWestCoeff` (`mtk::CoefficientFunction1D cw`)

Parameters

| | | |
|----|----|----------------------------------------------------------------------------|
| in | cw | Coeff. $c_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|----|----|----------------------------------------------------------------------------|

Definition at line 127 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



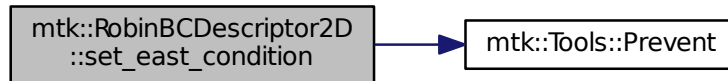
17.20.3.19 void mtk::RobinBCDescriptor2D::set_east_condition (Real(*) (const Real &yy, const Real &tt) east_condition)
[noexcept]

Parameters

| | | |
|----|----------------|------------------------------------------------------------------------|
| in | east_condition | $\beta_e(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$ |
|----|----------------|------------------------------------------------------------------------|

Definition at line 194 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



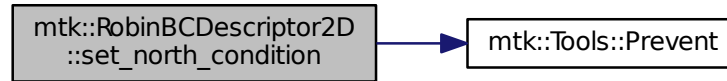
17.20.3.20 void mtk::RobinBCDescriptor2D::set_north_condition (Real(*) (const Real &xx, const Real &tt) north_condition)
[noexcept]

Parameters

| | | |
|----|-----------------|------------------------------------------------------------------------|
| in | north_condition | $\beta_n(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$ |
|----|-----------------|------------------------------------------------------------------------|

Definition at line 217 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



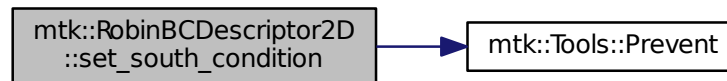
17.20.3.21 void mtk::RobinBCDescriptor2D::set_south_condition (Real(*) (const Real &xx, const Real &tt) south_condition)
[noexcept]

Parameters

| | | |
|----|-----------------|------------------------------------------------------------------------|
| in | south_condition | $\beta_s(x, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$ |
|----|-----------------|------------------------------------------------------------------------|

Definition at line 205 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



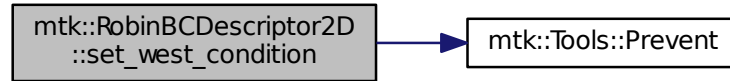
17.20.3.22 void mtk::RobinBCDescriptor2D::set_west_condition (Real(*) (const Real &yy, const Real &tt) west_condition)
[noexcept]

Parameters

| | | |
|----|----------------|------------------------------------------------------------------------|
| in | west_condition | $\beta_w(y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}.$ |
|----|----------------|------------------------------------------------------------------------|

Definition at line 183 of file [mtk_robin_bc_descriptor_2d.cc](#).

Here is the call graph for this function:



17.20.4 Member Data Documentation

17.20.4.1 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::east_coefficients_` [private]

Definition at line 367 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.2 `Real(* mtk::RobinBCDescriptor2D::east_condition_)(const Real &xx, const Real &tt)` [private]

Definition at line 372 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.3 `int mtk::RobinBCDescriptor2D::highest_order_diff_east_` [private]

Definition at line 362 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.4 `int mtk::RobinBCDescriptor2D::highest_order_diff_north_` [private]

Definition at line 364 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.5 `int mtk::RobinBCDescriptor2D::highest_order_diff_south_` [private]

Definition at line 363 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.6 `int mtk::RobinBCDescriptor2D::highest_order_diff_west_` [private]

Definition at line 361 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.7 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::north_coefficients_` [private]

Definition at line 369 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.8 `Real(* mtk::RobinBCDescriptor2D::north_condition_)(const Real &yy, const Real &tt)` [private]

Definition at line 374 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.9 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::south_coefficients_` [private]

Definition at line 368 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.10 `Real(* mtk::RobinBCDescriptor2D::south_condition_)(const Real &yy, const Real &tt)` [private]

Definition at line 373 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.11 `std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::west_coefficients_` [private]

Definition at line 366 of file [mtk_robin_bc_descriptor_2d.h](#).

17.20.4.12 `Real(* mtk::RobinBCDescriptor2D::west_condition_)(const Real &xx, const Real &tt)` [private]

Definition at line 371 of file [mtk_robin_bc_descriptor_2d.h](#).

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

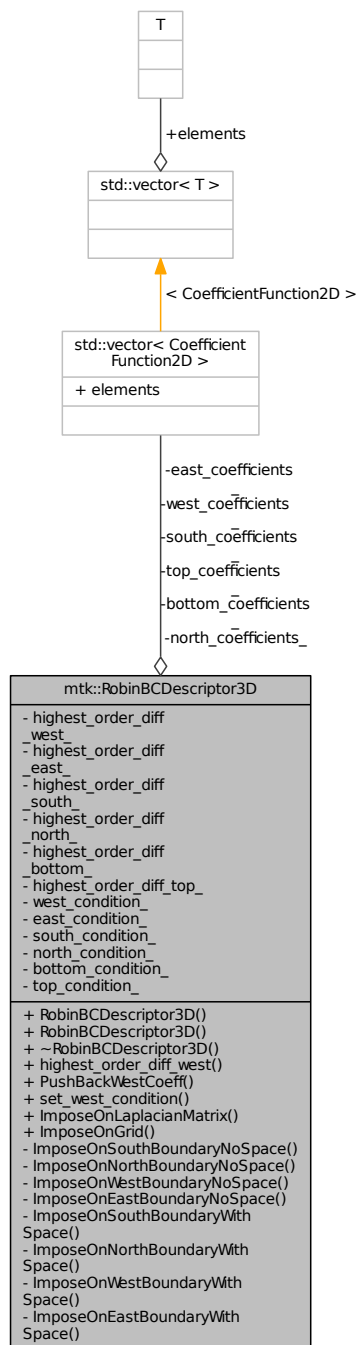
- [include/mtk_robin_bc_descriptor_2d.h](#)
- [src/mtk_robin_bc_descriptor_2d.cc](#)

17.21 mtk::RobinBCDescriptor3D Class Reference

Impose Robin boundary conditions on the operators and on the grids.

```
#include <mtk_robin_bc_descriptor_3d.h>
```

Collaboration diagram for mtk::RobinBCDescriptor3D:



Public Member Functions

- [RobinBCDescriptor3D \(\)](#)

Default constructor.

- [RobinBCDescriptor3D](#) (const [RobinBCDescriptor3D](#) &desc)

Copy constructor.

- [~RobinBCDescriptor3D](#) () noexcept

Destructor.

- int [highest_order_diff_west](#) () const noexcept

*Getter for highest order of differentiation in the * face.*

- void [PushBackWestCoeff](#) ([CoefficientFunction2D](#) cw)

Push back coefficient function at west lowest order diff. available.

- void [set_west_condition](#) ([Real](#)(*west_condition)(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)) noexcept

Set boundary condition at west.

- bool [ImposeOnLaplacianMatrix](#) (const [Lap3D](#) &lap, const [UniStgGrid3D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the operator represented as matrix.

- void [ImposeOnGrid](#) ([UniStgGrid3D](#) &grid, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the grid.

Private Member Functions

- bool [ImposeOnSouthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the south boundary.

- bool [ImposeOnNorthBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the north boundary.

- bool [ImposeOnWestBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the west boundary.

- bool [ImposeOnEastBoundaryNoSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the east boundary.

- bool [ImposeOnSouthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the south boundary.

- bool [ImposeOnNorthBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the north boundary.

- bool [ImposeOnWestBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the west boundary.

- bool [ImposeOnEastBoundaryWithSpace](#) (const [Lap2D](#) &lap, const [UniStgGrid2D](#) &grid, [DenseMatrix](#) &matrix, const [Real](#) &time=[kZero](#)) const

Imposes the condition on the east boundary.

Private Attributes

- int [highest_order_diff_west_](#)
Highest order of differentiation west.
- int [highest_order_diff_east_](#)
Highest order of differentiation east.
- int [highest_order_diff_south_](#)
Highest order differentiation for south.
- int [highest_order_diff_north_](#)
Highest order differentiation for north.
- int [highest_order_diff_bottom_](#)
Highest order differentiation bottom.
- int [highest_order_diff_top_](#)
Highest order differentiation for top.
- std::vector
< [CoefficientFunction2D](#) > [west_coefficients_](#)
Coeffs. west.
- std::vector
< [CoefficientFunction2D](#) > [east_coefficients_](#)
Coeffs. east.
- std::vector
< [CoefficientFunction2D](#) > [south_coefficients_](#)
Coeffs. south.
- std::vector
< [CoefficientFunction2D](#) > [north_coefficients_](#)
Coeffs. north.
- std::vector
< [CoefficientFunction2D](#) > [bottom_coefficients_](#)
Coeffs. bottom.
- std::vector
< [CoefficientFunction2D](#) > [top_coefficients_](#)
Coeffs. top.
- [Real](#)(* [west_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Condition west.
- [Real](#)(* [east_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Condition east.
- [Real](#)(* [south_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Cond. south.
- [Real](#)(* [north_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Cond. north.
- [Real](#)(* [bottom_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Cond. bottom.
- [Real](#)(* [top_condition_](#))(const [Real](#) &xx, const [Real](#) &yy, const [Real](#) &tt)
Cond. top.

17.21.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Definition at line 134 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.2 Constructor & Destructor Documentation

17.21.2.1 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D ()`

17.21.2.2 `mtk::RobinBCDescriptor3D::RobinBCDescriptor3D (const RobinBCDescriptor3D & desc)`

Parameters

| | | |
|-----------|-------------|----------------------|
| <i>in</i> | <i>desc</i> | Given 2D descriptor. |
|-----------|-------------|----------------------|

17.21.2.3 `mtk::RobinBCDescriptor3D::~~RobinBCDescriptor3D () [noexcept]`

17.21.3 Member Function Documentation

17.21.3.1 `int mtk::RobinBCDescriptor3D::highest_order_diff_west () const [noexcept]`

Returns

Integer highest order of differentiation in the * face.

17.21.3.2 `bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryNoSpace (const Lap2D & lap, const UniStgGrid2D & grid, DenseMatrix & matrix, const Real & time = kZero) const [private]`

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.3 **bool** mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryWithSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.4 **void** mtk::RobinBCDescriptor3D::ImposeOnGrid (UniStgGrid3D & *grid*, **const** Real & *time* = kZero) **const**

Parameters

| | | |
|---------|-------------|--------------------------------------------------------|
| in, out | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.5 **bool** mtk::RobinBCDescriptor3D::ImposeOnLaplacianMatrix (**const** Lap3D & *lap*, **const** UniStgGrid3D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero) **const**

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.6 **bool** mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryNoSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.7 **bool** mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryWithSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, DenseMatrix & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.8 **bool** mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryNoSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, **DenseMatrix** & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.9 **bool** mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryWithSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, **DenseMatrix** & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.10 **bool** mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryNoSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, **DenseMatrix** & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |
| in | <i>time</i> | Current time snapshot. Default is kZero. |

17.21.3.11 **bool** mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryWithSpace (**const** Lap2D & *lap*, **const** UniStgGrid2D & *grid*, **DenseMatrix** & *matrix*, **const** Real & *time* = kZero) **const** [private]

Parameters

| | | |
|---------|---------------|--------------------------------------------------------|
| in | <i>lap</i> | Laplacian operator on the matrix. |
| in | <i>grid</i> | Grid upon which impose the desired boundary condition. |
| in, out | <i>matrix</i> | Input matrix with the Laplacian operator. |

| | | |
|-----------|-------------|------------------------------------------|
| <i>in</i> | <i>time</i> | Current time snapshot. Default is kZero. |
|-----------|-------------|------------------------------------------|

17.21.3.12 void mtk::RobinBCDescriptor3D::PushBackWestCoeff (CoefficientFunction2D *cw*)

Parameters

| | | |
|-----------|-----------|-------------------------------------------------------------------------------|
| <i>in</i> | <i>cw</i> | Coeff. $c_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|-----------|-----------|-------------------------------------------------------------------------------|

17.21.3.13 void mtk::RobinBCDescriptor3D::set_west_condition (Real(*) (const Real &xx, const Real &yy, const Real &tt) *west_condition*) [noexcept]

Parameters

| | | |
|-----------|-----------------------|----------------------------------------------------------------------------|
| <i>in</i> | <i>west_condition</i> | $\beta_w(x, y, t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$. |
|-----------|-----------------------|----------------------------------------------------------------------------|

17.21.4 Member Data Documentation

17.21.4.1 std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::bottom_coefficients_ [private]

Definition at line 309 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.2 Real(* mtk::RobinBCDescriptor3D::bottom_condition_)(const Real &xx, const Real &yy, const Real &tt) [private]

Definition at line 324 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.3 std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::east_coefficients_ [private]

Definition at line 306 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.4 Real(* mtk::RobinBCDescriptor3D::east_condition_)(const Real &xx, const Real &yy, const Real &tt) [private]

Definition at line 315 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.5 int mtk::RobinBCDescriptor3D::highest_order_diff_bottom_ [private]

Definition at line 302 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.6 int mtk::RobinBCDescriptor3D::highest_order_diff_east_ [private]

Definition at line 299 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.7 int mtk::RobinBCDescriptor3D::highest_order_diff_north_ [private]

Definition at line 301 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.8 `int mtk::RobinBCDescriptor3D::highest_order_diff_south_` `[private]`

Definition at line 300 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.9 `int mtk::RobinBCDescriptor3D::highest_order_diff_top_` `[private]`

Definition at line 303 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.10 `int mtk::RobinBCDescriptor3D::highest_order_diff_west_` `[private]`

Definition at line 298 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.11 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::north_coefficients_` `[private]`

Definition at line 308 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.12 `Real(* mtk::RobinBCDescriptor3D::north_condition_)(const Real &xx, const Real &yy, const Real &tt)`
`[private]`

Definition at line 321 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.13 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::south_coefficients_` `[private]`

Definition at line 307 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.14 `Real(* mtk::RobinBCDescriptor3D::south_condition_)(const Real &xx, const Real &yy, const Real &tt)`
`[private]`

Definition at line 318 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.15 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::top_coefficients_` `[private]`

Definition at line 310 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.16 `Real(* mtk::RobinBCDescriptor3D::top_condition_)(const Real &xx, const Real &yy, const Real &tt)` `[private]`

Definition at line 327 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.17 `std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::west_coefficients_` `[private]`

Definition at line 305 of file [mtk_robin_bc_descriptor_3d.h](#).

17.21.4.18 `Real(* mtk::RobinBCDescriptor3D::west_condition_)(const Real &xx, const Real &yy, const Real &tt)`
`[private]`

Definition at line 312 of file [mtk_robin_bc_descriptor_3d.h](#).

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

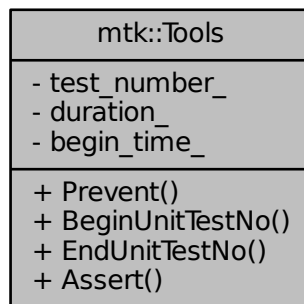
- [include/mtk_robin_bc_descriptor_3d.h](#)

17.22 mtk::Tools Class Reference

Tool manager class.

```
#include <mtk_tools.h>
```

Collaboration diagram for mtk::Tools:



Static Public Member Functions

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

Static Private Attributes

- static int [test_number_](#)
Current test being executed.

- static [Real duration_](#) {}
Duration of the current test.
- static clock_t [begin_time_](#) {}
Elapsed time on current test.

17.22.1 Detailed Description

Basic tools to ensure execution correctness, and to assists with unitary testing.

Definition at line 80 of file [mtk_tools.h](#).

17.22.2 Member Function Documentation

17.22.2.1 void [mtk::Tools::Assert](#) (const bool & *condition*) [static], [noexcept]

Parameters

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

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

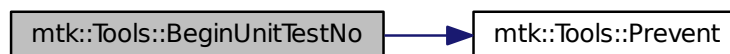
17.22.2.2 void [mtk::Tools::BeginUnitTestNo](#) (const int & *nn*) [static], [noexcept]

Parameters

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

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

Here is the call graph for this function:



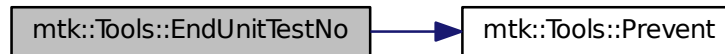
17.22.2.3 void [mtk::Tools::EndUnitTestNo](#) (const int & *nn*) [static], [noexcept]

Parameters

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

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

Here is the call graph for this function:



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

See also

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

Parameters

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

Todo Check if this is the best way of stalling execution.

Definition at line 62 of file [mtk_tools.cc](#).

17.22.3 Member Data Documentation

17.22.3.1 `clock_t mtk::Tools::begin_time_ {}` `[static], [private]`

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

17.22.3.2 `mtk::Real mtk::Tools::duration_ {}` `[static], [private]`

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

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

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

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

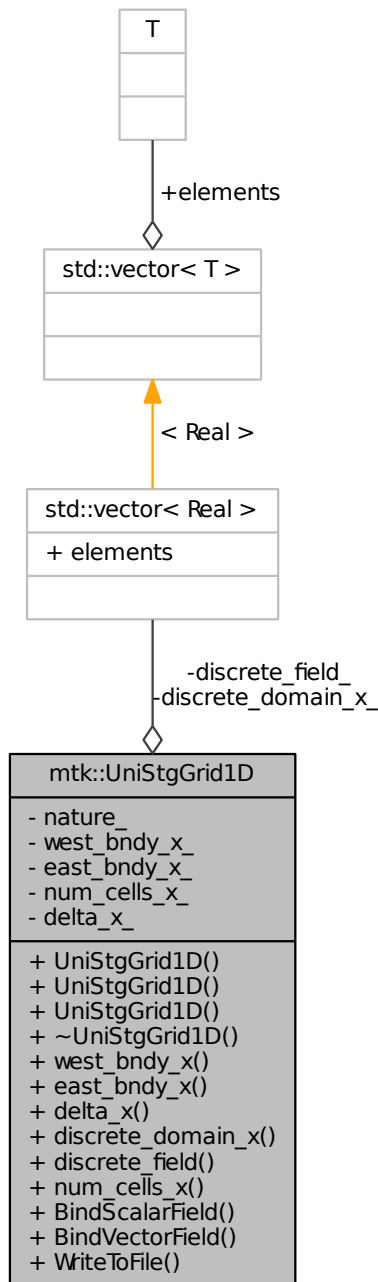
- [include/mtk_tools.h](#)
- [src/mtk_tools.cc](#)

17.23 mtk::UniStgGrid1D Class Reference

Uniform 1D Staggered Grid.

```
#include <mtk_uni_stg_grid_1d.h>
```

Collaboration diagram for mtk::UniStgGrid1D:



Public Member Functions

- [UniStgGrid1D](#) ()
Default constructor.
- [UniStgGrid1D](#) (const [UniStgGrid1D](#) &grid)
Copy constructor.
- [UniStgGrid1D](#) (const [Real](#) &west_bndy_x, const [Real](#) &east_bndy_x, const int &num_cells_x, const [mtk::Field](#) &nature=mtk::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 Δx .
- const [Real](#) * [discrete_domain_x](#) () const
Provides access to the grid spatial data.
- [Real](#) * [discrete_field](#) ()
Provides access to the grid field data.
- int [num_cells_x](#) () const
Provides access to the number of cells of the grid.
- void [BindScalarField](#) ([Real](#)(*ScalarField)(const [Real](#) &xx))
Binds a given scalar field to the grid.
- void [BindVectorField](#) ([Real](#)(*VectorField)([Real](#) xx))
Binds a given vector field to the grid.
- bool [WriteToFile](#) (std::string filename, std::string space_name, std::string field_name) const
Writes grid to a file compatible with gnuplot 4.6.

Private Attributes

- [FieldNature nature_](#)
Nature of the discrete field.
- std::vector< [Real](#) > [discrete_domain_x_](#)
Array of spatial data.
- std::vector< [Real](#) > [discrete_field_](#)
Array of field's data.
- [Real west_bndy_x_](#)
West boundary spatial coordinate.
- [Real east_bndy_x_](#)
East boundary spatial coordinate.
- [Real num_cells_x_](#)
Number of cells discretizing the domain.
- [Real delta_x_](#)
Produced Δx .

Friends

- `std::ostream & operator<< (std::ostream &stream, UniStgGrid1D &in)`
Prints the grid as a tuple of arrays.

17.23.1 Detailed Description

Uniform 1D Staggered Grid.

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

17.23.2 Constructor & Destructor Documentation

17.23.2.1 `mtk::UniStgGrid1D::UniStgGrid1D ()`

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

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

Parameters

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

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

17.23.2.3 `mtk::UniStgGrid1D::UniStgGrid1D (const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const mtk::FieldNature &nature = mtk::FieldNature::SCALAR)`

Parameters

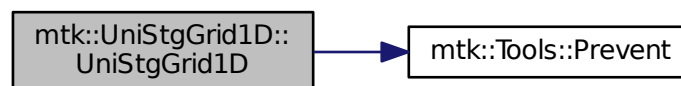
| | | |
|-----------------|--------------------------|---------------------------------------|
| <code>in</code> | <code>west_bndy_x</code> | Coordinate for the west boundary. |
| <code>in</code> | <code>east_bndy_x</code> | Coordinate for the east boundary. |
| <code>in</code> | <code>num_cells_x</code>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

See also

[mtk::FieldNature](#)

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

Here is the call graph for this function:



17.23.2.4 mtk::UniStgGrid1D::~~UniStgGrid1D ()

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

17.23.3 Member Function Documentation

17.23.3.1 void mtk::UniStgGrid1D::BindScalarField (*Real*(*) (const *Real* &xx) *ScalarField*)

Parameters

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

1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 176 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:

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

We assume the field to be of the form:

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

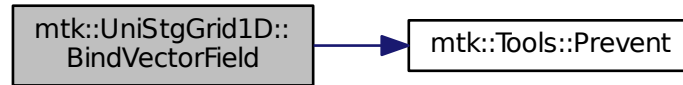
Parameters

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

1. Create collection of spatial coordinates.
2. Create collection of field samples.

Definition at line 213 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the call graph for this function:



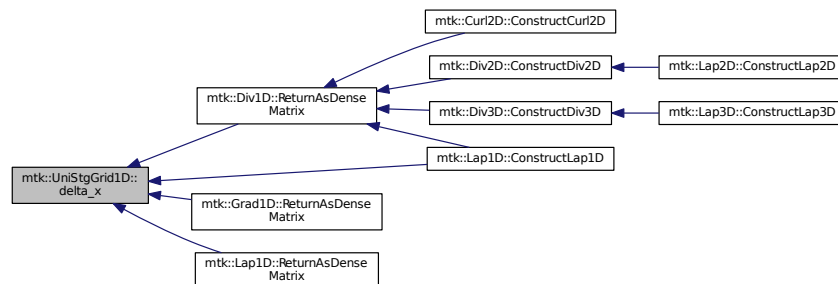
17.23.3.3 `mtk::Real mtk::UniStgGrid1D::delta_x () const`

Returns

Computed Δx .

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

Here is the caller graph for this function:



17.23.3.4 `const mtk::Real * mtk::UniStgGrid1D::discrete_domain_x () const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

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

17.23.3.5 `mtk::Real * mtk::UniStgGrid1D::discrete_field ()`

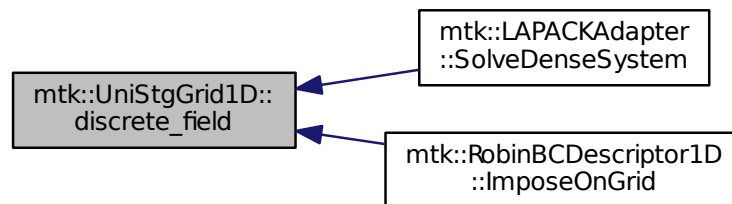
Returns

Pointer to the field data.

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

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

Here is the caller graph for this function:

**17.23.3.6 mtk::Real mtk::UniStgGrid1D::east_bndy_x () const****Returns**

East boundary spatial coordinate.

Definition at line 151 of file [mtk_uni_stg_grid_1d.cc](#).

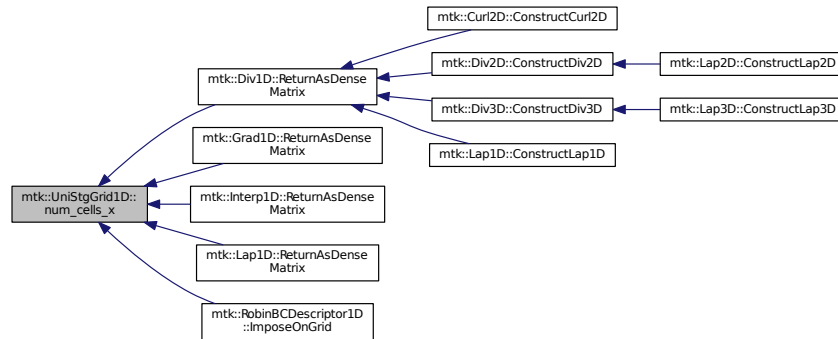
17.23.3.7 int mtk::UniStgGrid1D::num_cells_x () const

Returns

Number of cells of the grid.

Definition at line 171 of file [mtk_uni_stg_grid_1d.cc](#).

Here is the caller graph for this function:

**17.23.3.8 mtk::Real mtk::UniStgGrid1D::west_bndy_x () const****Returns**

West boundary spatial coordinate.

Definition at line 146 of file [mtk_uni_stg_grid_1d.cc](#).

17.23.3.9 bool mtk::UniStgGrid1D::WriteToFile (std::string filename, std::string space_name, std::string field_name) const**Parameters**

| | | |
|----|-------------------|-----------------------------------------|
| in | <i>filename</i> | Name of the output file. |
| in | <i>space_name</i> | Name for the first column of the data. |
| in | <i>field_name</i> | Name for the second column of the data. |

Returns

Success of the file writing process.

See also

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

Definition at line 242 of file [mtk_uni_stg_grid_1d.cc](#).

17.23.4 Friends And Related Function Documentation

17.23.4.1 `std::ostream& operator<< (std::ostream & stream, mtk::UniStgGrid1D & in)` [friend]

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

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

17.23.5 Member Data Documentation

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

Definition at line 199 of file [mtk_uni_stg_grid_1d.h](#).

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

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

17.23.5.3 `std::vector<Real>` `mtk::UniStgGrid1D::discrete_field_` [private]

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

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

Definition at line 197 of file [mtk_uni_stg_grid_1d.h](#).

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

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

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

Definition at line 198 of file [mtk_uni_stg_grid_1d.h](#).

17.23.5.7 **Real** `mtk::UniStgGrid1D::west_bndy_x_` [private]

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

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

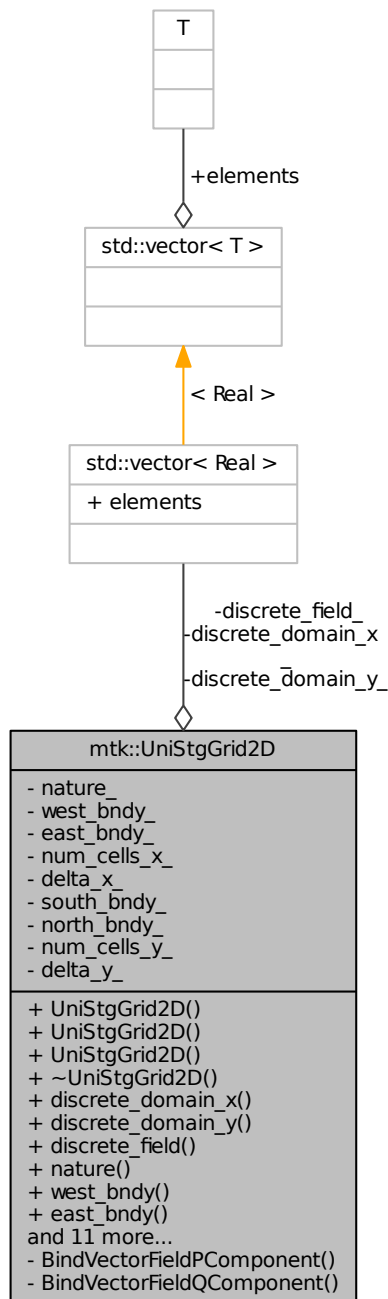
- [include/mtk_uni_stg_grid_1d.h](#)
- [src/mtk_uni_stg_grid_1d.cc](#)

17.24 mtk::UniStgGrid2D Class Reference

Uniform 2D Staggered Grid.

```
#include <mtk_uni_stg_grid_2d.h>
```

Collaboration diagram for `mtk::UniStgGrid2D`:



Public Member Functions

- [UniStgGrid2D](#) ()

Default constructor.

- **UniStgGrid2D** (const **UniStgGrid2D** &grid)

Copy constructor.

- **UniStgGrid2D** (const **Real** &west_bndy_x, const **Real** &east_bndy_x, const int &num_cells_x, const **Real** &south_bndy_y, const **Real** &north_bndy_y, const int &num_cells_y, const **mtk::FieldNature** &nature=**mtk::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 Δx .

- **Real** **south_bndy** () const

Provides access to south boundary spatial coordinate.

- **Real** **north_bndy** () const

Provides access to north boundary spatial coordinate.

- int **num_cells_y** () const

Provides access to the number of cells of the grid.

- **Real** **delta_y** () const

Provides access to the computed Δy .

- 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.

17.24.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 79 of file `mtk_uni_stg_grid_2d.h`.

17.24.2 Constructor & Destructor Documentation

17.24.2.1 mtk::UniStgGrid2D::UniStgGrid2D ()

Definition at line 132 of file [mtk_uni_stg_grid_2d.cc](#).

17.24.2.2 mtk::UniStgGrid2D::UniStgGrid2D (const UniStgGrid2D & grid)

Parameters

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

Definition at line 146 of file [mtk_uni_stg_grid_2d.cc](#).

17.24.2.3 mtk::UniStgGrid2D::UniStgGrid2D (const Real & west_bndy_x, const Real & east_bndy_x, const int & num_cells_x, const Real & south_bndy_y, const Real & north_bndy_y, const int & num_cells_y, const mtk::FieldNature & nature = mtk::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:



17.24.2.4 mtk::UniStgGrid2D::~~UniStgGrid2D ()

Definition at line 204 of file [mtk_uni_stg_grid_2d.cc](#).

17.24.3 Member Function Documentation

17.24.3.1 void mtk::UniStgGrid2D::BindScalarField (Real(*) (const Real &xx, const Real &yy) *ScalarField*)

Parameters

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

1. Create collection of spatial coordinates for x .
2. Create collection of spatial coordinates for y .
3. Create collection of field samples.

Definition at line 276 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



17.24.3.2 void mtk::UniStgGrid2D::BindVectorField (Real(*) (const Real &xx, const Real &yy) *VectorFieldPComponent*, Real(*) (const Real &xx, const Real &yy) *VectorFieldQComponent*)

We assume the field to be of the form:

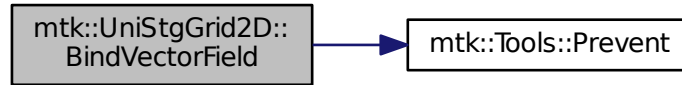
$$\mathbf{v}(\mathbf{x}) = p(x, y) \hat{\mathbf{i}} + q(x, y) \hat{\mathbf{j}}$$

Parameters

| | | |
|----|------------------------------|-----------------------------------------------------------------------------|
| in | <i>VectorFieldPComponent</i> | Pointer to the function implementing the p component of the vector field. |
| in | <i>VectorFieldQComponent</i> | Pointer to the function implementing the q component of the vector field. |

Definition at line 425 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the call graph for this function:



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

We assume the field to be of the form:

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

Parameters

| | | |
|----|----------------------------------|---------------------------------------------------------------------------------|
| in | <i>BindVectorFieldPComponent</i> | Pointer to the function implementing the \$ p \$ component of the vector field. |
|----|----------------------------------|---------------------------------------------------------------------------------|

1. Create collection of spatial coordinates for x .
2. Create collection of spatial coordinates for y .
3. Allocate space for discrete vector field and bind \$ p \$ component.

Definition at line 332 of file [mtk_uni_stg_grid_2d.cc](#).

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

We assume the field to be of the form:

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

Parameters

| | | |
|----|----------------------------------|---------------------------------------------------------------------------------|
| in | <i>BindVectorFieldQComponent</i> | Pointer to the function implementing the \$ q \$ component of the vector field. |
|----|----------------------------------|---------------------------------------------------------------------------------|

1. Bind \$ q \$ component, since \$ p \$ component has already been bound.

Definition at line 397 of file [mtk_uni_stg_grid_2d.cc](#).

17.24.3.5 bool mtk::UniStgGrid2D::Bound () const

Returns

True is a field has been bound.

Definition at line 256 of file [mtk_uni_stg_grid_2d.cc](#).

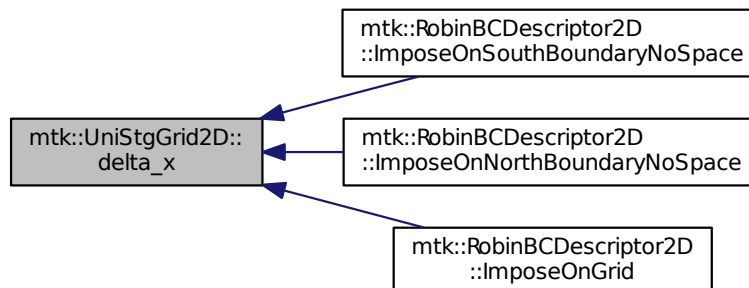
Here is the caller graph for this function:

**17.24.3.6 mtk::Real mtk::UniStgGrid2D::delta_x () const****Returns**

Computed Δx .

Definition at line 226 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:

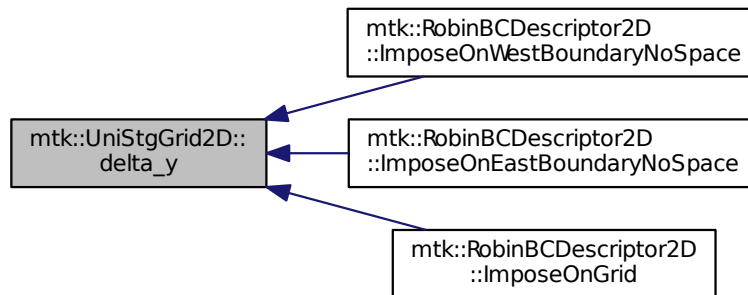
**17.24.3.7 mtk::Real mtk::UniStgGrid2D::delta_y () const**

Returns

Computed Δy .

Definition at line 251 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.24.3.8 `const mtk::Real * mtk::UniStgGrid2D::discrete_domain_x () const`

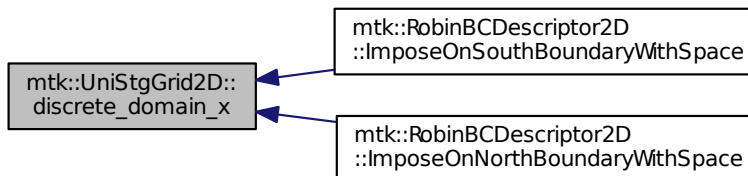
Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 231 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.24.3.9 `const mtk::Real * mtk::UniStgGrid2D::discrete_domain_y () const`

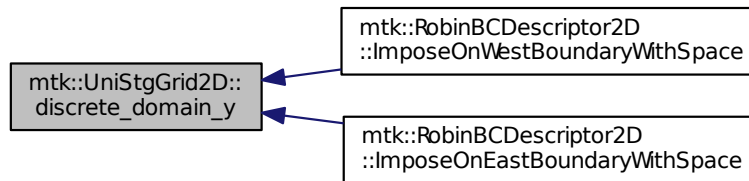
Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 261 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



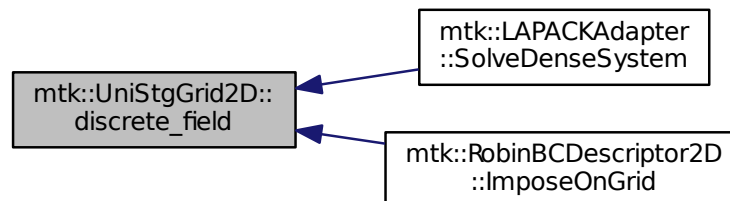
17.24.3.10 `mtk::Real * mtk::UniStgGrid2D::discrete_field ()`

Returns

Pointer to the field data.

Definition at line 266 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



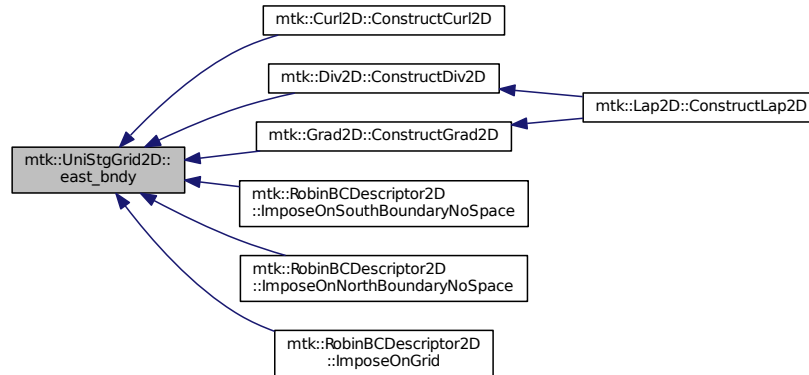
17.24.3.11 `mtk::Real mtk::UniStgGrid2D::east_bndy () const`

Returns

East boundary spatial coordinate.

Definition at line 216 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.24.3.12 mtk::FieldNature mtk::UniStgGrid2D::nature () const

Returns

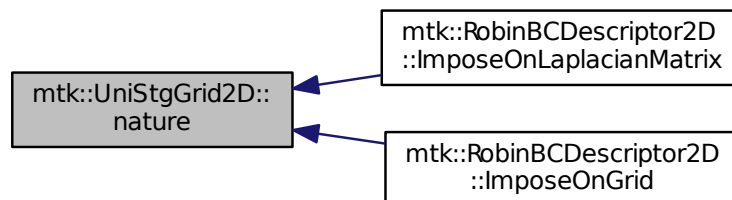
Value of an enumeration.

See also

[mtk::FieldNature](#)

Definition at line 206 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



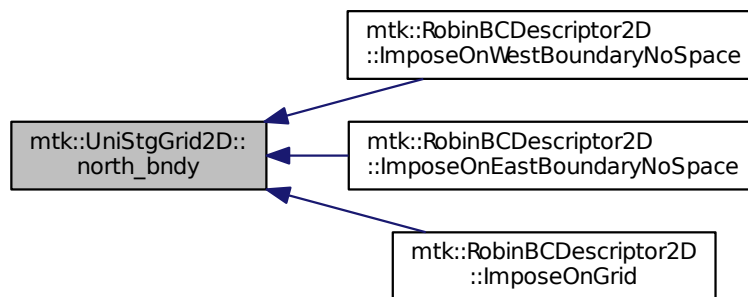
17.24.3.13 mtk::Real mtk::UniStgGrid2D::north_bndy () const

Returns

North boundary spatial coordinate.

Definition at line 241 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



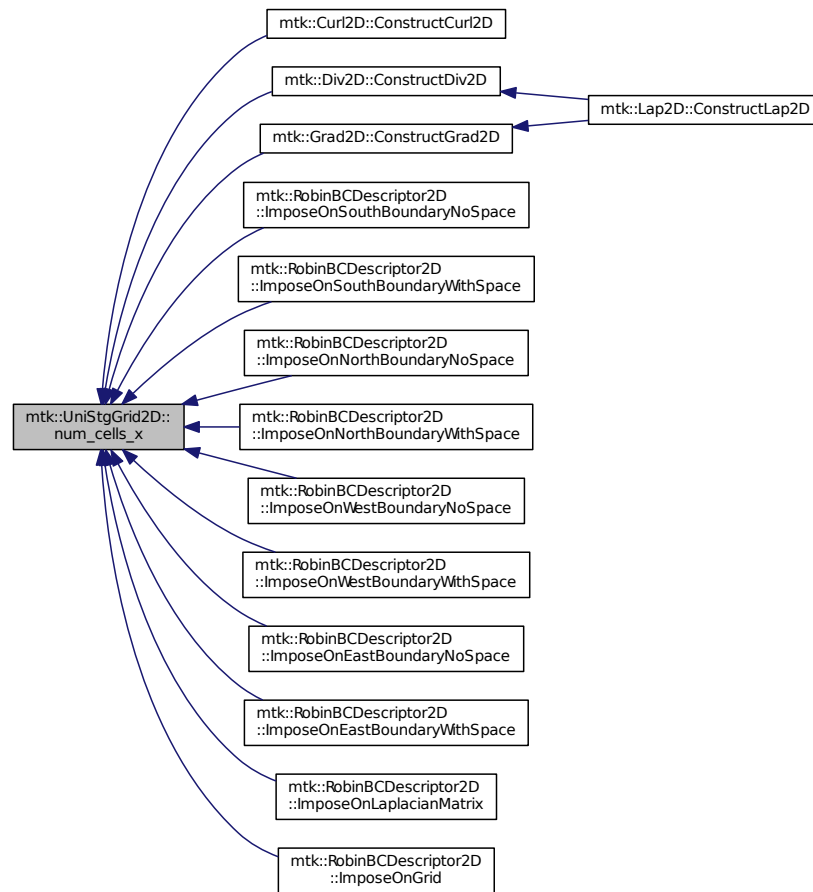
17.24.3.14 int mtk::UniStgGrid2D::num_cells_x () const

Returns

Number of cells of the grid.

Definition at line 221 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



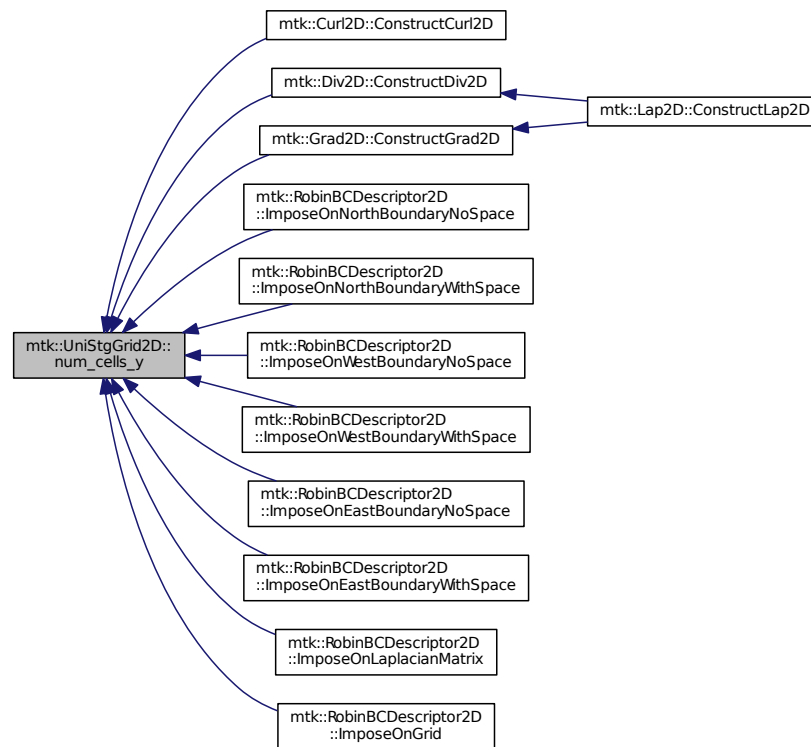
17.24.3.15 `int mtk::UniStgGrid2D::num_cells_y () const`

Returns

Number of cells of the grid.

Definition at line 246 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



17.24.3.16 `int mtk::UniStgGrid2D::Size () const`

Returns

Total number of samples in the grid.

Definition at line 271 of file [mtk_uni_stg_grid_2d.cc](#).

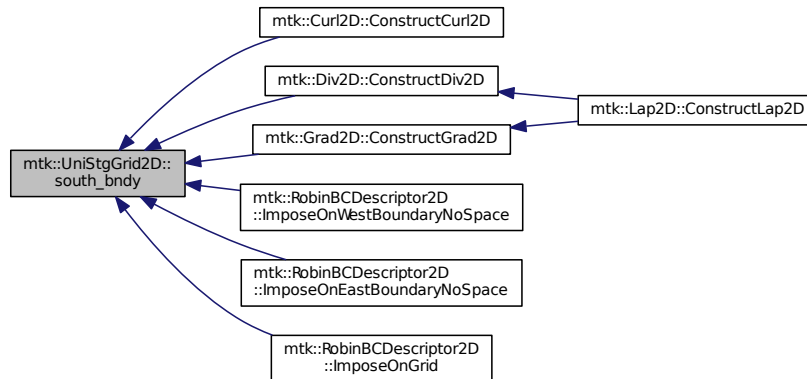
17.24.3.17 `mtk::Real mtk::UniStgGrid2D::south_bndy () const`

Returns

South boundary spatial coordinate.

Definition at line 236 of file [mtk_uni_stg_grid_2d.cc](#).

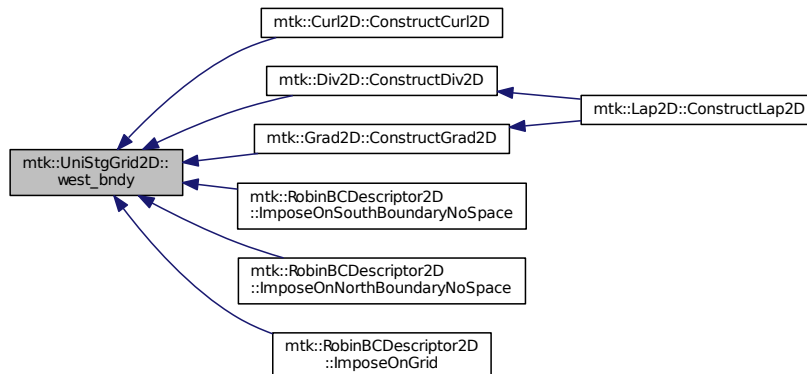
Here is the caller graph for this function:

**17.24.3.18 mtk::Real mtk::UniStgGrid2D::west_bndy () const****Returns**

West boundary spatial coordinate.

Definition at line 211 of file [mtk_uni_stg_grid_2d.cc](#).

Here is the caller graph for this function:



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

Parameters

| | | |
|----|---------------------|----------------------------------------------------------|
| in | <i>filename</i> | Name of the output file. |
| in | <i>space_name_x</i> | Name for the first column of the (spatial) data. |
| in | <i>space_name_y</i> | Name for the second column of the (spatial) data. |
| in | <i>field_name</i> | Name for the second column of the (physical field) data. |

Returns

Success of the file writing process.

See also

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

Write the values of the p component, with a null q component.

Write the values of the q component, with a null p component.

Definition at line 438 of file [mtk_uni_stg_grid_2d.cc](#).

17.24.4 Friends And Related Function Documentation

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

1. Print spatial coordinates.

2. Print scalar field.

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

17.24.5 Member Data Documentation

17.24.5.1 `Real mtk::UniStgGrid2D::delta_x_` [private]

Definition at line 302 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.2 `Real mtk::UniStgGrid2D::delta_y_` [private]

Definition at line 307 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 293 of file [mtk_uni_stg_grid_2d.h](#).

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

Definition at line 294 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.5 `std::vector<Real> mtk::UniStgGrid2D::discrete_field_` [private]

Definition at line 295 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.6 `Real mtk::UniStgGrid2D::east_bndy_` [private]

Definition at line 300 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.7 `FieldNature mtk::UniStgGrid2D::nature_` [private]

Definition at line 297 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.8 `Real mtk::UniStgGrid2D::north_bndy_` [private]

Definition at line 305 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.9 `int mtk::UniStgGrid2D::num_cells_x_` [private]

Definition at line 301 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.10 `int mtk::UniStgGrid2D::num_cells_y_` [private]

Definition at line 306 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.11 `Real mtk::UniStgGrid2D::south_bndy_` [private]

Definition at line 304 of file [mtk_uni_stg_grid_2d.h](#).

17.24.5.12 `Real mtk::UniStgGrid2D::west_bndy_` [private]

Definition at line 299 of file [mtk_uni_stg_grid_2d.h](#).

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

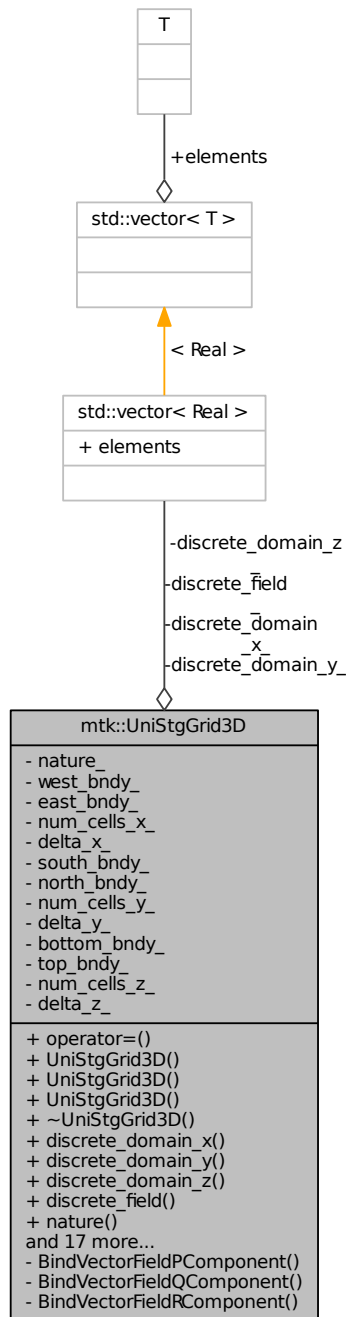
- [include/mtk_uni_stg_grid_2d.h](#)
- [src/mtk_uni_stg_grid_2d.cc](#)

17.25 mtk::UniStgGrid3D Class Reference

Uniform 3D Staggered Grid.

```
#include <mtk_uni_stg_grid_3d.h>
```


Collaboration diagram for mtk::UniStgGrid3D:



Public Member Functions

- [UniStgGrid3D operator=](#) (const [UniStgGrid3D](#) &in)

Overloaded assignment operator.

- [UniStgGrid3D](#) ()

Default constructor.

- [UniStgGrid3D](#) (const [UniStgGrid3D](#) &grid)

Copy constructor.

- [UniStgGrid3D](#) (const [Real](#) &west_bndy_x, const [Real](#) &east_bndy_x, const int &num_cells_x, const [Real](#) &south_bndy_y, const [Real](#) &north_bndy_y, const int &num_cells_y, const [Real](#) &bottom_bndy_z, const [Real](#) &top_bndy_z, const int &num_cells_z, const [mtk::FieldNature](#) &nature=[mtk::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 Δx .

- [Real](#) south_bndy () const

Provides access to south boundary spatial coordinate.

- [Real](#) north_bndy () const

Provides access to north boundary spatial coordinate.

- int num_cells_y () const

Provides access to the number of cells of the grid.

- [Real](#) delta_y () const

Provides access to the computed Δy .

- [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 Δz .

- 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_](#)

- East boundary spatial coordinate.*
- 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.

17.25.1 Detailed Description

Uniform 3D Staggered Grid.

Definition at line 79 of file [mtk_uni_stg_grid_3d.h](#).

17.25.2 Constructor & Destructor Documentation

17.25.2.1 `mtk::UniStgGrid3D::UniStgGrid3D ()`

Definition at line 123 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.2.2 `mtk::UniStgGrid3D::UniStgGrid3D (const UniStgGrid3D &grid)`

Parameters

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

Definition at line 142 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.2.3 `mtk::UniStgGrid3D::UniStgGrid3D (const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const Real &south_bndy_y, const Real &north_bndy_y, const int &num_cells_y, const Real &bottom_bndy_z, const Real &top_bndy_z, const int &num_cells_z, const mtk::FieldNature &nature = mtk::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:



17.25.2.4 mtk::UniStgGrid3D::~~UniStgGrid3D ()

Definition at line 221 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3 Member Function Documentation

17.25.3.1 void mtk::UniStgGrid3D::BindScalarField (Real(*) (const Real &xx, const Real &yy, const Real &zz) *ScalarField*)

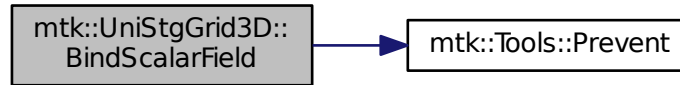
Parameters

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

1. Create collection of spatial coordinates for x .
2. Create collection of spatial coordinates for y .
3. Create collection of spatial coordinates for z .
4. Create collection of field samples.

Definition at line 318 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



17.25.3.2 void mtk::UniStgGrid3D::BindVectorField (Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldPComponent, Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldQComponent, Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldRComponent)

We assume the field to be of the form:

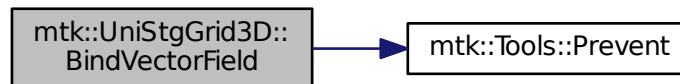
$$\mathbf{v}(\mathbf{x}) = p(x, y, z)\hat{\mathbf{i}} + q(x, y, z)\hat{\mathbf{j}} + r(x, y, z)\hat{\mathbf{k}}$$

Parameters

| | | |
|----|------------------------------|---------------------------------------------------------------------------------|
| in | <i>VectorFieldPComponent</i> | Pointer to the function implementing the \$ p \$ component of the vector field. |
| in | <i>VectorFieldQComponent</i> | Pointer to the function implementing the \$ q \$ component of the vector field. |
| in | <i>VectorFieldRComponent</i> | Pointer to the function implementing the \$ r \$ component of the vector field. |

Definition at line 415 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the call graph for this function:



17.25.3.3 void mtk::UniStgGrid3D::BindVectorFieldPComponent (Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldPComponent) [private]

We assume the field to be of the form:

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

Parameters

| | | |
|----|----------------------------------|---------------------------------------------------------------------------------|
| in | <i>BindVectorFieldPComponent</i> | Pointer to the function implementing the \$ p \$ component of the vector field. |
|----|----------------------------------|---------------------------------------------------------------------------------|

Definition at line 394 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.4 void mtk::UniStgGrid3D::BindVectorFieldQComponent (Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldQComponent) [private]

We assume the field to be of the form:

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

Parameters

| | | |
|----|----------------------------------|---------------------------------------------------------------------------------|
| in | <i>BindVectorFieldQComponent</i> | Pointer to the function implementing the \$ q \$ component of the vector field. |
|----|----------------------------------|---------------------------------------------------------------------------------|

Definition at line 401 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.5 void mtk::UniStgGrid3D::BindVectorFieldRComponent (Real(*) (const Real &xx, const Real &yy, const Real &zz) VectorFieldRComponent) [private]

We assume the field to be of the form:

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

Parameters

| | | |
|----|----------------------------------|---------------------------------------------------------------------------------|
| in | <i>BindVectorFieldRComponent</i> | Pointer to the function implementing the \$ r \$ component of the vector field. |
|----|----------------------------------|---------------------------------------------------------------------------------|

Definition at line 408 of file [mtk_uni_stg_grid_3d.cc](#).

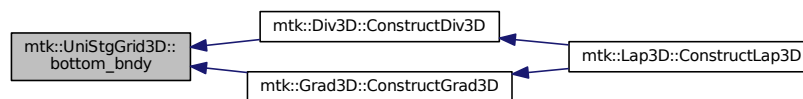
17.25.3.6 mtk::Real mtk::UniStgGrid3D::bottom_bndy () const

Returns

Bottom boundary spatial coordinate.

Definition at line 278 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.25.3.7 `bool mtk::UniStgGrid3D::Bound () const`

Returns

True is a field has been bound.

Definition at line 308 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.8 `mtk::Real mtk::UniStgGrid3D::delta_x () const`

Returns

Computed Δx .

Definition at line 243 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.9 `mtk::Real mtk::UniStgGrid3D::delta_y () const`

Returns

Computed Δy .

Definition at line 268 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.10 `mtk::Real mtk::UniStgGrid3D::delta_z () const`

Returns

Computed Δz .

Definition at line 293 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.11 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_x () const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 248 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.12 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_y () const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 273 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.13 `const mtk::Real * mtk::UniStgGrid3D::discrete_domain_z () const`

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 298 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.14 `mtk::Real * mtk::UniStgGrid3D::discrete_field ()`

Returns

Pointer to the field data.

Definition at line 303 of file [mtk_uni_stg_grid_3d.cc](#).

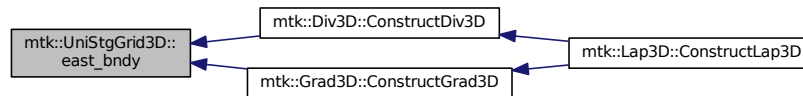
17.25.3.15 `mtk::Real mtk::UniStgGrid3D::east_bndy () const`

Returns

East boundary spatial coordinate.

Definition at line 233 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.25.3.16 `mtk::FieldNature mtk::UniStgGrid3D::nature () const`

Returns

Value of an enumeration.

See also

[mtk::FieldNature](#)

Definition at line 223 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.17 `mtk::Real mtk::UniStgGrid3D::north_bndy () const`

Returns

North boundary spatial coordinate.

Definition at line 258 of file [mtk_uni_stg_grid_3d.cc](#).

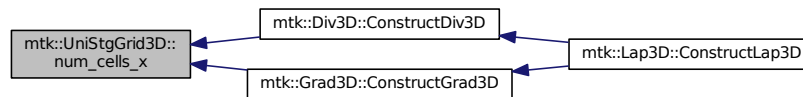
17.25.3.18 `int mtk::UniStgGrid3D::num_cells_x () const`

Returns

Number of cells of the grid.

Definition at line 238 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



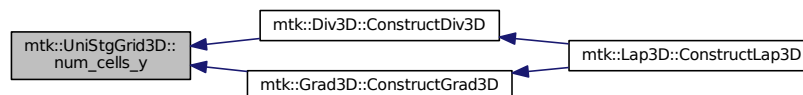
17.25.3.19 `int mtk::UniStgGrid3D::num_cells_y () const`

Returns

Number of cells of the grid.

Definition at line 263 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



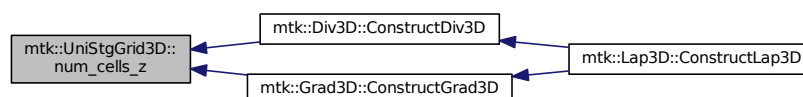
17.25.3.20 `int mtk::UniStgGrid3D::num_cells_z () const`

Returns

Number of cells of the grid.

Definition at line 288 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.25.3.21 `mtk::UniStgGrid3D mtk::UniStgGrid3D::operator= (const UniStgGrid3D & in)`

Parameters

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

Returns

Copy of the given grid.

Definition at line 116 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.3.22 `int mtk::UniStgGrid3D::Size () const`

Returns

Total number of samples in the grid.

Definition at line 313 of file [mtk_uni_stg_grid_3d.cc](#).

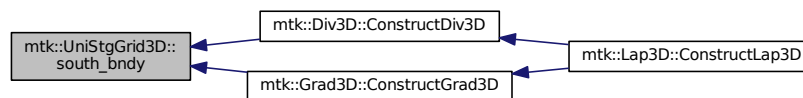
17.25.3.23 `mtk::Real mtk::UniStgGrid3D::south_bndy () const`

Returns

South boundary spatial coordinate.

Definition at line 253 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



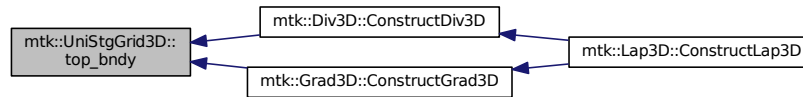
17.25.3.24 `mtk::Real mtk::UniStgGrid3D::top_bndy () const`

Returns

Top boundary spatial coordinate.

Definition at line 283 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



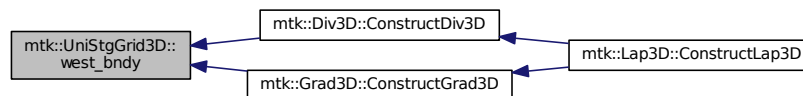
17.25.3.25 `mtk::Real mtk::UniStgGrid3D::west_bndy () const`

Returns

West boundary spatial coordinate.

Definition at line 228 of file [mtk_uni_stg_grid_3d.cc](#).

Here is the caller graph for this function:



17.25.3.26 `bool mtk::UniStgGrid3D::WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string space_name_z, std::string field_name) const`

Parameters

| | | |
|----|---------------------|----------------------------------------------------------|
| in | <i>filename</i> | Name of the output file. |
| in | <i>space_name_x</i> | Name for the first column of the (spatial) data. |
| in | <i>space_name_y</i> | Name for the second column of the (spatial) data. |
| in | <i>space_name_z</i> | Name for the third column of the (spatial) data. |
| in | <i>field_name</i> | Name for the second column of the (physical field) data. |

Returns

Success of the file writing process.

See also

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

Definition at line 435 of file [mtk_uni_stg_grid_3d.cc](#).

17.25.4 Friends And Related Function Documentation

17.25.4.1 `std::ostream& operator<< (std::ostream & stream, mtk::UniStgGrid3D & in)` [friend]

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

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

17.25.5 Member Data Documentation

17.25.5.1 `Real mtk::UniStgGrid3D::bottom_bndy_` [private]

Definition at line 396 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.2 `Real mtk::UniStgGrid3D::delta_x_` [private]

Definition at line 389 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.3 `Real mtk::UniStgGrid3D::delta_y_` [private]

Definition at line 394 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.4 `Real mtk::UniStgGrid3D::delta_z_` [private]

Definition at line 399 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.5 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_x_` [private]

Definition at line 379 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.6 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_y_` [private]

Definition at line 380 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.7 `std::vector<Real> mtk::UniStgGrid3D::discrete_domain_z_` [private]

Definition at line 381 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.8 `std::vector<Real> mtk::UniStgGrid3D::discrete_field_` [private]

Definition at line 382 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.9 `Real mtk::UniStgGrid3D::east_bndy_` [private]

Definition at line 387 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.10 `FieldNature mtk::UniStgGrid3D::nature_` [private]

Definition at line 384 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.11 `Real mtk::UniStgGrid3D::north_bndy_` [private]

Definition at line 392 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.12 `int mtk::UniStgGrid3D::num_cells_x_` [private]

Definition at line 388 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.13 `int mtk::UniStgGrid3D::num_cells_y_` [private]

Definition at line 393 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.14 `int mtk::UniStgGrid3D::num_cells_z_` [private]

Definition at line 398 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.15 `Real mtk::UniStgGrid3D::south_bndy_` [private]

Definition at line 391 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.16 `Real mtk::UniStgGrid3D::top_bndy_` [private]

Definition at line 397 of file [mtk_uni_stg_grid_3d.h](#).

17.25.5.17 `Real mtk::UniStgGrid3D::west_bndy_` [private]

Definition at line 386 of file [mtk_uni_stg_grid_3d.h](#).

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

- [include/mtk_uni_stg_grid_3d.h](#)
- [src/mtk_uni_stg_grid_3d.cc](#)

Chapter 18

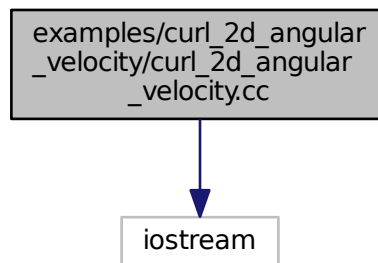
File Documentation

18.1 examples/curl_2d_angular_velocity/curl_2d_angular_velocity.cc File Reference

Compute the curl of a 2D angular velocity field.

```
#include <iostream>
```

Include dependency graph for curl_2d_angular_velocity.cc:



Functions

- int `main` ()

18.1.1 Detailed Description

We compute the curl of:

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

Author

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

Definition in file [curl_2d_angular_velocity.cc](#).

18.1.2 Function Documentation**18.1.2.1 int main ()**

Definition at line 106 of file [curl_2d_angular_velocity.cc](#).

18.2 curl_2d_angular_velocity.cc

```
00001
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #if __cplusplus == 201103L
00060
00061 #include <iostream>
00062 #include <fstream>
00063 #include <cmath>
00064
00065 #include <vector>
00066
00067 #include "mtk.h"
00068
```



```

00069 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const
    mtk::Real &yy) {
00070
00071     return -yy;
00072 }
00073
00074 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
    mtk::Real &yy) {
00075
00076     return xx;
00077 }
00078
00079 int main () {
00080
00081     std::cout << "Example: Curl of a angular velocity field." << std::endl;
00082
00083     mtk::Real aa = 0.0;
00084     mtk::Real bb = 4.0;
00085     mtk::Real cc = 0.0;
00086     mtk::Real dd = 4.0;
00087
00088     int nn = 10;
00089     int mm = 10;
00090
00091     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm,
    mtk::FieldNature::VECTOR);
00092
00093     gg.BindVectorField(VectorFieldPComponent, VectorFieldQComponent);
00094
00095     if(!gg.WriteToFile("curl_2d_angular_velocity_gg.dat", "x", "y", "v(x,y)")) {
00096         std::cerr << "Angular field could not be written to disk." << std::endl;
00097         return EXIT_FAILURE;
00098     }
00099 }
00100 }
00101
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

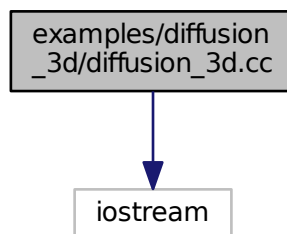
```

18.3 examples/diffusion_3d/diffusion_3d.cc File Reference

Diffusion Equation on a 3D Uniform Staggered Grid with Dirichlet BCs.

```
#include <iostream>
```

Include dependency graph for diffusion_3d.cc:



Functions

- `int main ()`

18.3.1 Detailed Description

We solve:

$$\frac{\partial u}{\partial t} = \nabla^2 u(\mathbf{x}),$$

for $\mathbf{x} \in \Omega = [0, 1]^3$.

We consider autonomous homogeneous Dirichlet boundary conditions.

Author

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

Definition in file [diffusion_3d.cc](#).

18.3.2 Function Documentation

18.3.2.1 `int main ()`

Definition at line 123 of file [diffusion_3d.cc](#).

18.4 `diffusion_3d.cc`

```

00001
00016 /*
00017 Copyright (C) 2015, Computational Science Research Center, San Diego State
00018 University. All rights reserved.
00019
00020 Redistribution and use in source and binary forms, with or without modification,
00021 are permitted provided that the following conditions are met:
00022
00023 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00024 and a copy of the modified files should be reported once modifications are
00025 completed, unless these modifications are made through the project's GitHub
00026 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00027 should be developed and included in any deliverable.
00028
00029 2. Redistributions of source code must be done through direct
00030 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00031
00032 3. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 4. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders, and due credit should
00038 be given to the copyright holders.
00039
00040 5. Neither the name of the copyright holder nor the names of its contributors
00041 may be used to endorse or promote products derived from this software without
00042 specific prior written permission.
00043
00044 The copyright holders provide no reassurances that the source code provided does
00045 not infringe any patent, copyright, or any other intellectual property rights of
00046 third parties. The copyright holders disclaim any liability to any recipient for
00047 claims brought against recipient by any third party for infringement of that
00048 parties intellectual property rights.
00049
00050 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

```

```

00051 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00052 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00053 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00054 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00055 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00056 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00057 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00058 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00059 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00060 */
00061
00062 #if __cplusplus == 201103L
00063
00064 #include <iostream>
00065 #include <fstream>
00066 #include <cmath>
00067
00068 #include <vector>
00069
00070 #include "mtk.h"
00071
00072 mtk::Real InitialCondition(const mtk::Real &xx,
00073                             const mtk::Real &yy,
00074                             const mtk::Real &zz) {
00075
00076     mtk::Real rr{0.3};
00077
00078     mtk::Real aux{xx*xx + yy*yy + zz*zz};
00079
00080     return (aux < rr? rr - aux: mtk::kZero);
00081 }
00082
00083 int main () {
00084
00085     std::cout << "Example: Diffusion Equation in 3D "
00086                 "with Dirichlet BCs." << std::endl;
00087
00088     mtk::Real west_bndy_x{0.0};
00089     mtk::Real east_bndy_x{1.0};
00090     mtk::Real south_bndy_y{0.0};
00091     mtk::Real north_bndy_y{1.0};
00092     mtk::Real bottom_bndy_z{0.0};
00093     mtk::Real top_bndy_z{1.0};
00094
00095     int num_cells_x{50};
00096     int num_cells_y{50};
00097     int num_cells_z{50};
00098
00099     mtk::UniStgGrid3D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00100                                south_bndy_y, north_bndy_y, num_cells_y,
00101                                bottom_bndy_z, top_bndy_z, num_cells_z);
00102
00103     comp_sol.BindScalarField(InitialCondition);
00104
00105     if(!comp_sol.WriteToFile("diffusion_3d_comp_sol.dat",
00106                              "x",
00107                              "y",
00108                              "z",
00109                              "Initial u(x,y,z)")) {
00110         std::cerr << "Error writing to file." << std::endl;
00111         return EXIT_FAILURE;
00112     }
00113
00114 #else
00115 #include <iostream>
00116 using std::cout;
00117 using std::endl;
00118 int main () {
00119     cout << "This code HAS to be compiled with support for C++11." << endl;
00120     cout << "Exiting..." << endl;
00121     return EXIT_SUCCESS;
00122 }
00123 #endif

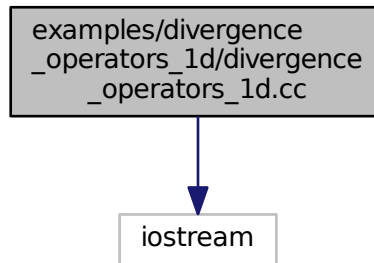
```

18.5 examples/divergence_operators_1d/divergence_operators_1d.cc File Reference

Creates instances of a 1D divergence as computed by the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for divergence_operators_1d.cc:



Functions

- int [main](#) ()

18.5.1 Detailed Description

Author

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

Definition in file [divergence_operators_1d.cc](#).

18.5.2 Function Documentation

18.5.2.1 int main ()

Definition at line [102](#) of file [divergence_operators_1d.cc](#).

18.6 divergence_operators_1d.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
  
```

```

00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Instances of a 1D divergence as computed by the CBS "
00067         "algorithm." << std::endl;
00068
00069     std::ofstream output_tex_file;
00070
00071     int max_order{14};
00072
00073     for (int order = 2; order <= max_order; order += 2) {
00074
00075         std::string output_tex_file_name{"div_1d_" + std::to_string(order) +
00076             ".tex"};
00077
00078         output_tex_file.open(output_tex_file_name);
00079
00080         mtk::Div1D div;
00081
00082         bool assertion = div.ConstructDiv1D(order);
00083
00084         if (!assertion) {
00085             std::cerr << "Mimetic div (order" + std::to_string(order) +
00086                 ") could not be built." << std::endl;
00087             return EXIT_FAILURE;
00088         }
00089
00090         output_tex_file << "\\begin{verbatim}" << std::endl;
00091         output_tex_file << div << std::endl;
00092         output_tex_file << "\\end{verbatim}" << std::endl;
00093         output_tex_file.close();
00094     }
00095 }
00096
00097 #else
00098

```

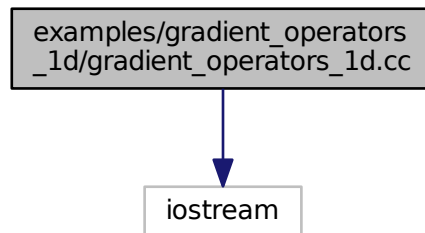
```
00099 #include <iostream>
00100 using std::cout;
00101 using std::endl;
00102 int main () {
00103     cout << "This code HAS to be compiled with support for C++11." << endl;
00104     cout << "Exiting..." << endl;
00105     return EXIT_SUCCESS;
00106 }
00107 #endif
```

18.7 examples/gradient_operators_1d/gradient_operators_1d.cc File Reference

Creates instances of a 1D gradient as computed by the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for gradient_operators_1d.cc:



Functions

- `int main ()`

18.7.1 Detailed Description

Author

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

Definition in file [gradient_operators_1d.cc](#).

18.7.2 Function Documentation

18.7.2.1 `int main ()`

Definition at line [102](#) of file [gradient_operators_1d.cc](#).

18.8 gradient_operators_1d.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Instances of a 1D gradient as computed by the CBS "
00067         "algorithm." << std::endl;
00068
00069
00070
00071     std::ofstream output_tex_file;
00072
00073     int max_order{14};
00074
00075     for (int order = 2; order <= max_order; order += 2) {
00076
00077         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

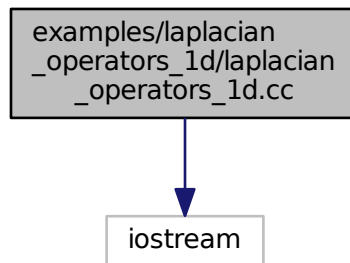
```

18.9 examples/laplacian_operators_1d/laplacian_operators_1d.cc File Reference

Creates instances of a 1D Laplacian as computed by the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for laplacian_operators_1d.cc:



Functions

- int [main](#) ()

18.9.1 Detailed Description

Author

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

Definition in file [laplacian_operators_1d.cc](#).

18.9.2 Function Documentation

18.9.2.1 int main ()

Definition at line 102 of file [laplacian_operators_1d.cc](#).

18.10 laplacian_operators_1d.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <string>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Instances of a 1D Laplacian as computed by the CBS "
00067         "algorithm." << std::endl;
00068
00069
00070
00071     std::ofstream output_tex_file;
00072
00073     int max_order{14};
00074

```

```

00075     for (int order = 2; order <= max_order; order += 2) {
00076
00077         std::string output_tex_file_name{"lap_1d_" + std::to_string(order) +
00078             ".tex"};
00079
00080         output_tex_file.open(output_tex_file_name);
00081
00082         mtk::Lap1D lap;
00083
00084         bool assertion = lap.ConstructLap1D(order);
00085         if (!assertion) {
00086             std::cerr << "Mimetic lap (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 << lap << 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

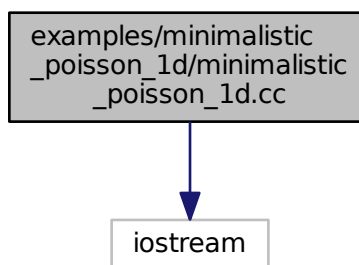
```

18.11 examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for minimalistic_poisson_1d.cc:



Functions

- int `main` ()

18.11.1 Detailed Description

We solve:

$$-\nabla^2 p(x) = s(x),$$

for $x \in \Omega = [a, b] = [0, 1]$.

The source term function is defined as:

$$s(x) = -\frac{\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1},$$

where $\lambda = -1$ is a real-valued parameter.

We consider Robin's boundary conditions of the form:

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

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

where $\alpha = -\exp(\lambda)$, $\beta = (\exp(\lambda) - 1.0)/\lambda$, $\omega = -1$, and $\varepsilon = 0$.

The analytical solution for this problem is given by:

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

The mimetic counterpart of this equation is:

$$-\mathbf{\check{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

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

18.11.2 Function Documentation

18.11.2.1 int main ()

Definition at line 164 of file [minimalistic_poisson_1d.cc](#).

18.12 minimalistic_poisson_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk

```

```

00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #if __cplusplus == 201103L
00090
00091 #include <iostream>
00092 #include <fstream>
00093 #include <cmath>
00094 #include <vector>
00095
00096 #include "mtk.h"
00097
00098 mtk::Real Alpha(const mtk::Real &tt) {
00099     mtk::Real lambda = -1.0;
00100     return -exp(lambda);
00101 }
00102
00103 mtk::Real Beta(const mtk::Real &tt) {
00104     mtk::Real lambda = -1.0;
00105     return (exp(lambda) - 1.0)/lambda;
00106 };
00107
00108 mtk::Real Omega(const mtk::Real &tt) { return -1.0; };
00109
00110 mtk::Real Epsilon(const mtk::Real &tt) { return 0.0; };
00111
00112 mtk::Real Source(const mtk::Real &xx) {
00113     mtk::Real lambda = -1.0;
00114     return lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00115 }
00116
00117 mtk::Real KnownSolution(const mtk::Real &xx) {
00118     mtk::Real lambda = -1.0;
00119     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00120 }
00121
00122 int main () {
00123
00124     mtk::Real west_bndy_x{};
00125     mtk::Real east_bndy_x{1.0};
00126     int num_cells_x{5};
00127     mtk::Lap1D lap;
00128     if (!lap.ConstructLap1D()) {
00129         return EXIT_FAILURE;
00130     }
00131     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00132     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00133     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00134     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00135     source.BindScalarField(Source);
00136     mtk::RobinBCDescriptor1D bcs;
00137     bcs.PushBackWestCoeff(Alpha);
00138     bcs.PushBackWestCoeff(Beta);

```

```

00139 bcs.PushBackEastCoeff(Alpha);
00140 bcs.PushBackEastCoeff(Beta);
00141 bcs.set_west_condition(Omega);
00142 bcs.set_east_condition(Epsilon);
00143 if (!bcs.ImposeOnLaplacianMatrix(lap, lapm)) {
00144     return EXIT_FAILURE;
00145 }
00146 bcs.ImposeOnGrid(source);
00147 int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00148 if (info != 0) {
00149     return EXIT_FAILURE;
00150 }
00151 source.WriteToFile("minimalistic_poisson_1d_comp_sol.dat", "x", "~u(x)");
00152 known_sol.BindScalarField(KnownSolution);
00153 known_sol.WriteToFile("minimalistic_poisson_1d_known_sol.dat", "x", "u(x)");
00154 mtk::Real relative_norm_2_error =
00155     mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00156                                     known_sol.discrete_field(),
00157                                     known_sol.num_cells_x());
00158 std::cout << relative_norm_2_error << std::endl;
00159 }
00160 #else
00161 #include <iostream>
00162 using std::cout;
00163 using std::endl;
00164 int main () {
00165     cout << "This code HAS to be compiled with support for C++11." << endl;
00166     cout << "Exiting..." << endl;
00167     return EXIT_SUCCESS;
00168 }
00169 #endif

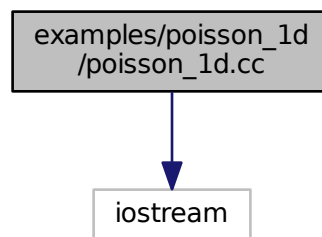
```

18.13 examples/poisson_1d/poisson_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for poisson_1d.cc:



Functions

- int `main` ()

18.13.1 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:

$$-\check{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

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

18.13.2 Function Documentation

18.13.2.1 int main ()

Definition at line 263 of file [poisson_1d.cc](#).

18.14 poisson_1d.cc

```
00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
```

```

00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #if __cplusplus == 201103L
00090
00091 #include <iostream>
00092 #include <fstream>
00093 #include <cmath>
00094
00095 #include <vector>
00096
00097 #include "mtk.h"
00098
00099 mtk::Real Alpha(const mtk::Real &tt) {
00100
00101     mtk::Real lambda{-1.0};
00102
00103     return -exp(lambda);
00104 }
00105
00106 mtk::Real Beta(const mtk::Real &tt) {
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) {
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) {
00131
00132     mtk::Real lambda{-1.0};
00133
00134     return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00135 }
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     mtk::Real west_bndy_x{0.0};
00143     mtk::Real east_bndy_x{1.0};
00144     int num_cells_x{50};
00145
00146     mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00147
00148     mtk::Lapl1D lap;
00149
00150     if (!lap.ConstructLapl1D()) {
00151         std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00152         return EXIT_FAILURE;
00153     }
00154
00155     std::cout << "lap=" << std::endl;
00156     std::cout << lap << std::endl;
00157
00158     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00159
00160     std::cout << "lapm =" << std::endl;
00161     std::cout << lapm << std::endl;
00162
00163     lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00164
00165     std::cout << "-lapm =" << std::endl;
00166     std::cout << lapm << std::endl;
00167
00168     mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00169
00170     source.BindScalarField(Source);
00171
00172     std::cout << "source =" << std::endl;
00173     std::cout << source << std::endl;
00174
00175     mtk::RobinBCDescriptor1D robin_bc_desc_ld;
00176
00177     robin_bc_desc_ld.PushBackWestCoeff(Alpha);
00178     robin_bc_desc_ld.PushBackWestCoeff(Beta);
00179
00180     robin_bc_desc_ld.PushBackEastCoeff(Alpha);
00181     robin_bc_desc_ld.PushBackEastCoeff(Beta);
00182
00183     robin_bc_desc_ld.set_west_condition(Omega);
00184     robin_bc_desc_ld.set_east_condition(Epsilon);
00185
00186     if (!robin_bc_desc_ld.ImposeOnLaplacianMatrix(lap, lapm)) {
00187         std::cerr << "BCs could not be bound to the matrix." << std::endl;
00188         return EXIT_FAILURE;
00189     }
00190
00191     std::cout << "Mimetic Laplacian operator with imposed BCs:" << std::endl;
00192     std::cout << lapm << std::endl;
00193
00194     if (!lapm.WriteToFile("poisson_ld_lapm.dat")) {
00195         std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00196         return EXIT_FAILURE;
00197     }
00198
00199     robin_bc_desc_ld.ImposeOnGrid(source);
00200
00201     std::cout << "source =" << std::endl;
00202     std::cout << source << std::endl;
00203
00204     if (!source.WriteToFile("poisson_ld_source.dat", "x", "s(x)")) {
00205         std::cerr << "Source term could not be written to disk." << std::endl;
00206         return EXIT_FAILURE;
00207     }
00208
00209     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00210
00211     if (!info) {
00212         std::cout << "System solved." << std::endl;
00213         std::cout << std::endl;
00214     } else {
00215         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00216         std::cerr << "Exiting..." << std::endl;
00217         return EXIT_FAILURE;
00218     }
00219

```



```

00227
00228     std::cout << "Computed solution:" << std::endl;
00229     std::cout << source << std::endl;
00230
00231     if (!source.WriteToFile("poisson_1d_comp_sol.dat", "x", "~u(x)")) {
00232         std::cerr << "Solution could not be written to file." << std::endl;
00233         return EXIT_FAILURE;
00234     }
00235
00237     mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00238
00239     known_sol.BindScalarField(KnownSolution);
00240
00241     std::cout << "known_sol =" << std::endl;
00242     std::cout << known_sol << std::endl;
00243
00244     if (!known_sol.WriteToFile("poisson_1d_known_sol.dat", "x", "u(x)")) {
00245         std::cerr << "Known solution could not be written to file." << std::endl;
00246         return EXIT_FAILURE;
00247     }
00248
00249     mtk::Real relative_norm_2_error{};
00250
00251     relative_norm_2_error =
00252         mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00253                                         known_sol.discrete_field(),
00254                                         known_sol.num_cells_x());
00255
00256     std::cout << "relative_norm_2_error =" << std::endl;
00257     std::cout << relative_norm_2_error << std::endl;
00258 }
00259 #else
00260 #include <iostream>
00261 using std::cout;
00262 using std::endl;
00263 int main () {
00264     cout << "This code HAS to be compiled with support for C++11." << endl;
00265     cout << "Exiting..." << endl;
00266     return EXIT_SUCCESS;
00267 }
00268 #endif

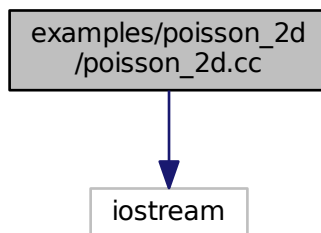
```

18.15 examples/poisson_2d/poisson_2d.cc File Reference

Poisson Equation on a 2D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```

Include dependency graph for poisson_2d.cc:



Functions

- `int main ()`

18.15.1 Detailed Description

We solve:

$$\nabla^2 u(\mathbf{x}) = s(\mathbf{x}),$$

for $\mathbf{x} \in \Omega = [0, 1]^2$.

The source term function is defined as

$$s(x, y) = xye^{-0.5(x^2+y^2)}(x^2 + y^2 - 6).$$

Let $\partial\Omega = S \cup N \cup W \cup E$. We consider Dirichlet boundary conditions of the following form:

$$\forall \mathbf{x} \in W : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in E : u(1, y) = -e^{-0.5(1-y^2)}(1 - y^2).$$

$$\forall \mathbf{x} \in S : u(\mathbf{x}) = 0.$$

$$\forall \mathbf{x} \in N : u(x, 1) = -e^{-0.5(x^2-1)}(x^2 - 1).$$

The analytical solution for this problem is given by

$$u(x, y) = xye^{-0.5(x^2+y^2)}.$$

Author

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

Definition in file [poisson_2d.cc](#).

18.15.2 Function Documentation

18.15.2.1 `int main ()`

Definition at line [241](#) of file [poisson_2d.cc](#).

18.16 `poisson_2d.cc`

```
00001
00039 /*
00040 Copyright (C) 2015, Computational Science Research Center, San Diego State
00041 University. All rights reserved.
00042
00043 Redistribution and use in source and binary forms, with or without modification,
00044 are permitted provided that the following conditions are met:
00045
00046 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00047 and a copy of the modified files should be reported once modifications are
00048 completed, unless these modifications are made through the project's GitHub
00049 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00050 should be developed and included in any deliverable.
00051
```

```

00052 2. Redistributions of source code must be done through direct
00053 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00054
00055 3. Redistributions in binary form must reproduce the above copyright notice,
00056 this list of conditions and the following disclaimer in the documentation and/or
00057 other materials provided with the distribution.
00058
00059 4. Usage of the binary form on proprietary applications shall require explicit
00060 prior written permission from the the copyright holders, and due credit should
00061 be given to the copyright holders.
00062
00063 5. Neither the name of the copyright holder nor the names of its contributors
00064 may be used to endorse or promote products derived from this software without
00065 specific prior written permission.
00066
00067 The copyright holders provide no reassurances that the source code provided does
00068 not infringe any patent, copyright, or any other intellectual property rights of
00069 third parties. The copyright holders disclaim any liability to any recipient for
00070 claims brought against recipient by any third party for infringement of that
00071 parties intellectual property rights.
00072
00073 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00074 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00075 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00076 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00077 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00078 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00079 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00080 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00081 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00082 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00083 */
00084
00085 #if __cplusplus == 201103L
00086
00087 #include <iostream>
00088 #include <fstream>
00089 #include <cmath>
00090
00091 #include <vector>
00092
00093 #include "mtk.h"
00094
00095 mtk::Real Source(const mtk::Real &xx, const mtk::Real &yy) {
00096
00097     mtk::Real x_squared{xx*xx};
00098     mtk::Real y_squared{yy*yy};
00099     mtk::Real aux{-0.5*(x_squared + y_squared)};
00100
00101     return xx*yy*exp(aux)*(x_squared + y_squared - 6.0);
00102 }
00103
00104 mtk::Real BCCoeff(const mtk::Real &xx, const mtk::Real &yy) {
00105
00106     return mtk::kOne;
00107 }
00108
00109 mtk::Real WestBC(const mtk::Real &xx, const mtk::Real &tt) {
00110
00111     return mtk::kZero;
00112 }
00113
00114 mtk::Real EastBC(const mtk::Real &yy, const mtk::Real &tt) {
00115
00116     return yy*exp(-0.5*(mtk::kOne + yy*yy));
00117 }
00118
00119 mtk::Real SouthBC(const mtk::Real &xx, const mtk::Real &tt) {
00120
00121     return mtk::kZero;
00122 }
00123
00124 mtk::Real NorthBC(const mtk::Real &xx, const mtk::Real &tt) {
00125
00126     return xx*exp(-0.5*(xx*xx + mtk::kOne));
00127 }
00128
00129 mtk::Real KnownSolution(const mtk::Real &xx, const mtk::Real &yy) {
00130
00131     mtk::Real x_squared{xx*xx};
00132     mtk::Real y_squared{yy*yy};

```

```

00133     mtk::Real aux{-0.5*(x_squared + y_squared)};
00134
00135     return xx*yy*exp(aux);
00136 }
00137
00138 int main () {
00139
00140     std::cout << "Example: Poisson Equation on a 2D Uniform Staggered Grid ";
00141     std::cout << "with Dirichlet and Neumann BCs." << std::endl;
00142
00143     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{5};
00148     int num_cells_y{5};
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     mtk::Lap2D lap;
00154
00155     if (!lap.ConstructLap2D(comp_sol)) {
00156         std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00157         return EXIT_FAILURE;
00158     }
00159
00160     mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix());
00161
00162     mtk::UniStgGrid2D source(west_bndy_x, east_bndy_x, num_cells_x,
00163                              south_bndy_y, north_bndy_y, num_cells_y);
00164
00165     source.BindScalarField(Source);
00166
00167     mtk::RobinBCDescriptor2D bcd;
00168
00169     bcd.PushBackWestCoeff(BCCoeff);
00170     bcd.PushBackEastCoeff(BCCoeff);
00171     bcd.PushBackSouthCoeff(BCCoeff);
00172     bcd.PushBackNorthCoeff(BCCoeff);
00173
00174     bcd.ImposeOnLaplacianMatrix(lap, comp_sol, lapm);
00175
00176     if (!lapm.WriteToFile("poisson_2d_lapm.dat")) {
00177         std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00178         return EXIT_FAILURE;
00179     }
00180
00181     bcd.set_west_condition(WestBC);
00182     bcd.set_east_condition(EastBC);
00183     bcd.set_south_condition(SouthBC);
00184     bcd.set_north_condition(NorthBC);
00185
00186     bcd.ImposeOnGrid(source);
00187
00188     if(!source.WriteToFile("poisson_2d_source.dat", "x", "y", "s(x,y)")) {
00189         std::cerr << "Source term could not be written to disk." << std::endl;
00190         return EXIT_FAILURE;
00191     }
00192
00193     int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00194
00195     if (!info) {
00196         std::cout << "System solved." << std::endl;
00197         std::cout << std::endl;
00198     } else {
00199         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00200         std::cerr << "Exiting..." << std::endl;
00201         return EXIT_FAILURE;
00202     }
00203
00204     if (!source.WriteToFile("poisson_2d_comp_sol.dat", "x", "y", "~u(x,y)")) {
00205         std::cerr << "Solution could not be written to file." << std::endl;
00206         return EXIT_FAILURE;
00207     }
00208
00209     mtk::UniStgGrid2D known_sol(west_bndy_x, east_bndy_x, num_cells_x,
00210                                 south_bndy_y, north_bndy_y, num_cells_y);
00211
00212     known_sol.BindScalarField(KnownSolution);
00213
00214
00215
00216
00217
00218
00219
00220

```

```

00221  if (!known_sol.WriteToFile("poisson_2d_known_sol.dat", "x", "y", "u(x,y)")) {
00222      std::cerr << "Known solution could not be written to file." << std::endl;
00223      return EXIT_FAILURE;
00224  }
00225
00226  mtk::Real relative_norm_2_error{};
00227
00228  relative_norm_2_error =
00229      mtk::BLASAdapter::RelNorm2Error(source.discrete_field(),
00230                                     known_sol.discrete_field(),
00231                                     known_sol.Size());
00232
00233  std::cout << "relative_norm_2_error = ";
00234  std::cout << relative_norm_2_error << std::endl;
00235 }
00236
00237 #else
00238 #include <iostream>
00239 using std::cout;
00240 using std::endl;
00241 int main () {
00242     cout << "This code HAS to be compiled with support for C++11." << endl;
00243     cout << "Exiting..." << endl;
00244     return EXIT_SUCCESS;
00245 }
00246 #endif

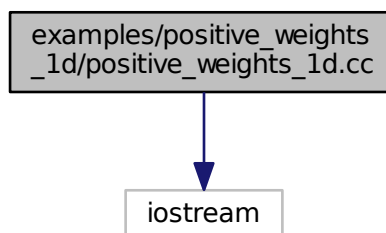
```

18.17 examples/positive_weights_1d/positive_weights_1d.cc File Reference

The CBS algorithm computes positive-definite weights, for 1D operators.

```
#include <iostream>
```

Include dependency graph for positive_weights_1d.cc:



Functions

- int [main](#) ()

18.17.1 Detailed Description

Author

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

Definition in file [positive_weights_1d.cc](#).

18.17.2 Function Documentation

18.17.2.1 `int main ()`

Definition at line 118 of file [positive_weights_1d.cc](#).

18.18 `positive_weights_1d.cc`

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <fstream>
00058 #include <cmath>
00059
00060 #include <vector>
00061
00062 #include "mtk.h"
00063
00064 int main () {
00065
00066     std::cout << "Example: Positive-Definite Weights for 1D Mimetic"
00067         "Operators." << std::endl;
00068
00069
00070
00071     mtk::Grad1D grad10;
00072
00073     bool assertion = grad10.ConstructGrad1D(10);
00074     if (!assertion) {

```

```

00075     std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00076     return EXIT_FAILURE;
00077 }
00078
00079 mtk::Grad1D grad12;
00080
00081 assertion = grad12.ConstructGrad1D(12);
00082 if (!assertion) {
00083     std::cerr << "Mimetic grad (12th order) could not be built." << std::endl;
00084     return EXIT_FAILURE;
00085 }
00086
00088
00089 mtk::Div1D div8;
00090
00091 assertion = div8.ConstructDiv1D(8);
00092 if (!assertion) {
00093     std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00094     return EXIT_FAILURE;
00095 }
00096
00097 mtk::Div1D div10;
00098
00099 assertion = div10.ConstructDiv1D(10);
00100 if (!assertion) {
00101     std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00102     return EXIT_FAILURE;
00103 }
00104
00105 mtk::Div1D div12;
00106
00107 assertion = div12.ConstructDiv1D(12);
00108 if (!assertion) {
00109     std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00110     return EXIT_FAILURE;
00111 }
00112 }
00113
00114 #else
00115 #include <iostream>
00116 using std::cout;
00117 using std::endl;
00118 int main () {
00119     cout << "This code HAS to be compiled with support for C++11." << endl;
00120     cout << "Exiting..." << endl;
00121     return EXIT_SUCCESS;
00122 }
00123 #endif

```

18.19 include/mtk.h File Reference

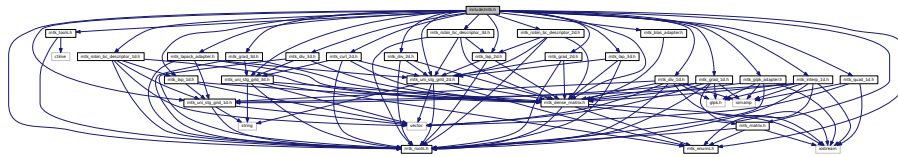
Includes the entire API.

```

#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_matrix.h"
#include "mtk_dense_matrix.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
#include "mtk_robin_bc_descriptor_1d.h"
#include "mtk_quad_1d.h"
#include "mtk_interp_1d.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"
#include "mtk_lap_2d.h"
#include "mtk_robin_bc_descriptor_2d.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"
#include "mtk_robin_bc_descriptor_3d.h"

```

Include dependency graph for mtk.h:



18.19.1 Detailed Description

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

Warning

It is extremely important that the headers are added to this file in a specific order; that is, considering the dependence between the classes these contain.

Author

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

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

18.20 mtk.h

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00277 #ifndef MTK_INCLUDE_MTK_H_
00278 #define MTK_INCLUDE_MTK_H_
00279
00287 #include "mtk_roots.h"
00288
00296 #include "mtk_enums.h"
00297
00305 #include "mtk_tools.h"
00306
00314 #include "mtk_matrix.h"
00315 #include "mtk_dense_matrix.h"
00316
00324 #include "mtk_blas_adapter.h"
00325 #include "mtk_lapack_adapter.h"
00326 #include "mtk_glpk_adapter.h"
00327
00335 #include "mtk_uni_stg_grid_1d.h"
00336 #include "mtk_uni_stg_grid_2d.h"
00337 #include "mtk_uni_stg_grid_3d.h"
00338
00346 #include "mtk_grad_1d.h"
00347 #include "mtk_div_1d.h"
00348 #include "mtk_lap_1d.h"
00349 #include "mtk_robin_bc_descriptor_1d.h"
00350 #include "mtk_quad_1d.h"
00351 #include "mtk_interp_1d.h"
00352
00353 #include "mtk_grad_2d.h"
00354 #include "mtk_div_2d.h"
00355 #include "mtk_curl_2d.h"
00356 #include "mtk_lap_2d.h"
00357 #include "mtk_robin_bc_descriptor_2d.h"
```

```

00358
00359 #include "mtk_grad_3d.h"
00360 #include "mtk_div_3d.h"
00361 #include "mtk_lap_3d.h"
00362 #include "mtk_robin_bc_descriptor_3d.h"
00363
00364 #endif // End of: MTK_INCLUDE_MTK_H_

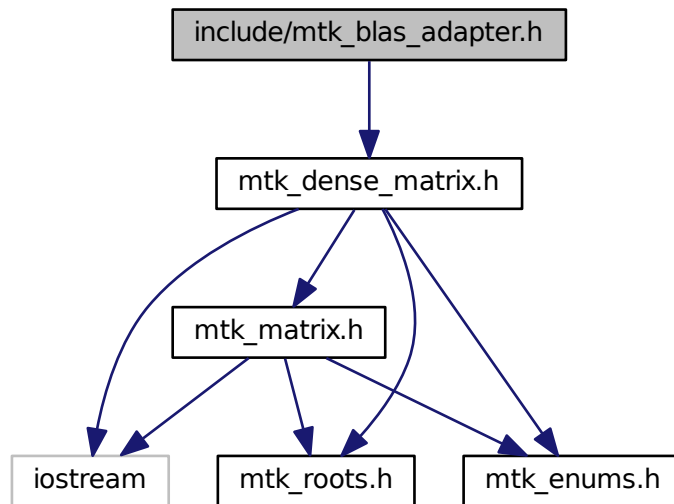
```

18.21 include/mtk_blas_adapter.h File Reference

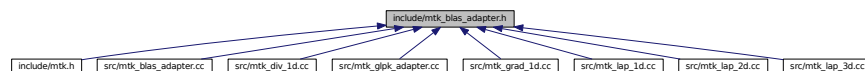
Adapter class for the BLAS API.

```
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk_blas_adapter.h:



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



Classes

- class `mtk::BLASAdapter`

Adapter class for the BLAS API.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.21.1 Detailed Description

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

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

The BLAS can be installed from links given in the See Also section of this page.

See also

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

Author

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

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

18.22 mtk_blas_adapter.h

```

00001
00025 /*
00026 Copyright (C) 2015, Computational Science Research Center, San Diego State
00027 University. All rights reserved.
00028
00029 Redistribution and use in source and binary forms, with or without modification,
00030 are permitted provided that the following conditions are met:
00031
00032 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00033 and a copy of the modified files should be reported once modifications are
00034 completed, unless these modifications are made through the project's GitHub
00035 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00036 should be developed and included in any deliverable.
00037
00038 2. Redistributions of source code must be done through direct
00039 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00040
00041 3. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 4. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders, and due credit should
00047 be given to the copyright holders.
00048
00049 5. Neither the name of the copyright holder nor the names of its contributors
00050 may be used to endorse or promote products derived from this software without
00051 specific prior written permission.
00052
00053 The copyright holders provide no reassurances that the source code provided does
00054 not infringe any patent, copyright, or any other intellectual property rights of
00055 third parties. The copyright holders disclaim any liability to any recipient for
00056 claims brought against recipient by any third party for infringement of that
00057 parties intellectual property rights.

```

```

00058
00059 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00060 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00061 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00062 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00063 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00064 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00065 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00066 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00067 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00068 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00069 */
00070
00071 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00072 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00073
00074 #include "mtk_dense_matrix.h"
00075
00076 namespace mtk {
00077
00099 class BLASAdapter {
00100 public:
00109     static Real RealNRM2(Real *in, int &in_length);
00110
00127     static void RealAXPY(Real alpha, Real *xx, Real *yy, int &in_length);
00128
00143     static Real RelNorm2Error(Real *computed, Real *known, int length);
00144
00162     static void RealDenseMV(Real &alpha,
00163                             DenseMatrix &aa,
00164                             Real *xx,
00165                             Real &beta,
00166                             Real *yy);
00167
00182     static DenseMatrix RealDenseMM(DenseMatrix &aa,
DenseMatrix &bb);
00183
00198     static DenseMatrix RealDenseSM(Real alpha,
DenseMatrix &aa);
00199 };
00200 }
00201 #endif // End of: MTK_INCLUDE_BLAS_ADAPTER_H_

```

18.23 include/mtk_curl_2d.h File Reference

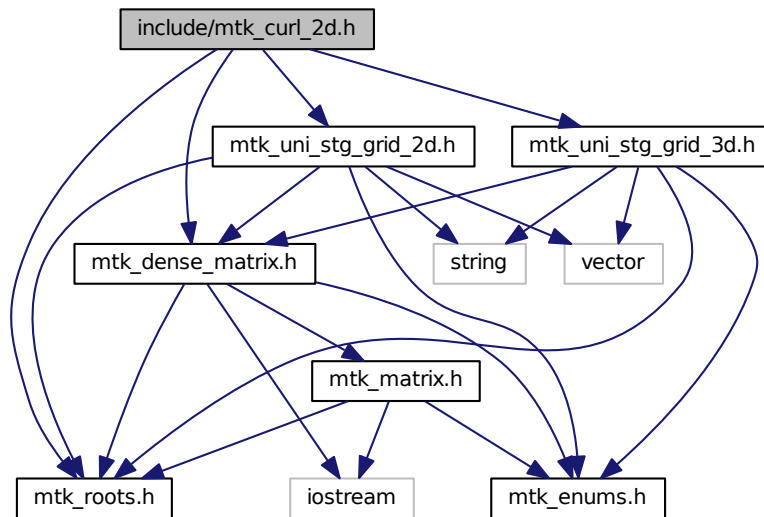
Includes the definition of the class Curl2D.

```

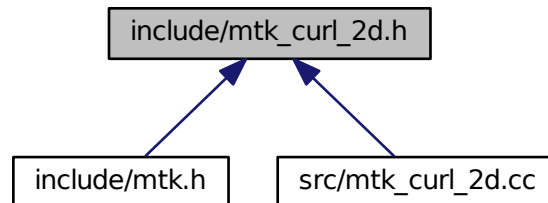
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for mtk_curl_2d.h:



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



Classes

- class [mtk::Curl2D](#)
Implements a 2D mimetic curl operator.

Namespaces

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

18.23.1 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](#).

18.24 mtk_curl_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_CURL_2D_H_
00058 #define MTK_INCLUDE_MTK_CURL_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk{
00066
00077 class Curl2D {
00078 public:
00080     UniStgGrid3D operator*(const UniStgGrid2D &grid) const;
00081

```

```

00083  Curl2D();
00084
00090  Curl2D(const Curl2D &curl);
00091
00093  ~Curl2D();
00094
00100  bool ConstructCurl2D(const UniStgGrid2D &grid,
00101                      int order_accuracy = kDefaultOrderAccuracy,
00102                      Real mimetic_threshold = kDefaultMimeticThreshold);
00103
00109  DenseMatrix ReturnAsDenseMatrix() const;
00110
00111 private:
00112  DenseMatrix curl_;
00113
00114  int order_accuracy_;
00115
00116  Real mimetic_threshold_;
00117 };
00118 }
00119 #endif // End of: MTK_INCLUDE_MTK_CURL_2D_H_

```

18.25 include/mtk_dense_matrix.h File Reference

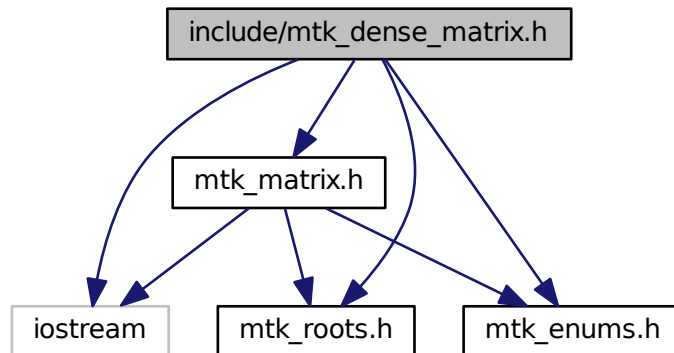
Defines a common dense matrix, using a 1D array.

```

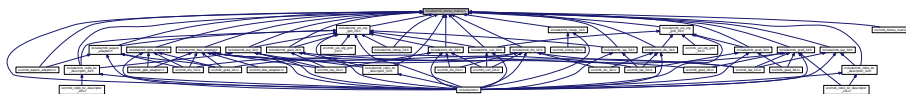
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_matrix.h"

```

Include dependency graph for mtk_dense_matrix.h:



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



Classes

- class [mtk::DenseMatrix](#)

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

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.25.1 Detailed Description

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

Author

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

Note

We prefer composition to inheritance [Reedy, 2011]. The main reason for this preference is that inheritance produces a more tightly coupled design. When a class inherits from another type be it public, protected, or private inheritance the subclass gains access to all public and protected members of the base class, whereas with composition, the class is only coupled to the public members of the other class. Furthermore, if you only hold a pointer to the other object, then your interface can use a forward declaration of the class rather than `#include` its full definition. This results in greater compile-time insulation and improves the time it takes to compile your code.

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

18.26 mtk_dense_matrix.h

```
00001
00023 /*
00024 Copyright (C) 2015, Computational Science Research Center, San Diego State
00025 University. All rights reserved.
00026
00027 Redistribution and use in source and binary forms, with or without modification,
00028 are permitted provided that the following conditions are met:
00029
00030 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00031 and a copy of the modified files should be reported once modifications are
00032 completed, unless these modifications are made through the project's GitHub
00033 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00034 should be developed and included in any deliverable.
00035
00036 2. Redistributions of source code must be done through direct
00037 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00038
00039 3. Redistributions in binary form must reproduce the above copyright notice,
00040 this list of conditions and the following disclaimer in the documentation and/or
00041 other materials provided with the distribution.
00042
00043 4. Usage of the binary form on proprietary applications shall require explicit
00044 prior written permission from the the copyright holders, and due credit should
00045 be given to the copyright holders.
00046
00047 5. Neither the name of the copyright holder nor the names of its contributors
```



```

00048 may be used to endorse or promote products derived from this software without
00049 specific prior written permission.
00050
00051 The copyright holders provide no reassurances that the source code provided does
00052 not infringe any patent, copyright, or any other intellectual property rights of
00053 third parties. The copyright holders disclaim any liability to any recipient for
00054 claims brought against recipient by any third party for infringement of that
00055 parties intellectual property rights.
00056
00057 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00058 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00059 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00060 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00061 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00062 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00063 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00064 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00065 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00066 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00067 */
00068
00069 #ifndef MTK_INCLUDE_DENSE_MATRIX_H_
00070 #define MTK_INCLUDE_DENSE_MATRIX_H_
00071
00072 #include <iostream>
00073
00074 #include "mtk_roots.h"
00075 #include "mtk_enums.h"
00076 #include "mtk_matrix.h"
00077
00078 namespace mtk {
00079
00092 class DenseMatrix {
00093 public:
00095     friend std::ostream& operator <<(std::ostream &stream, DenseMatrix &in);
00096
00104     DenseMatrix& operator =(const DenseMatrix &in);
00105
00107     bool operator ==(const DenseMatrix &in);
00108
00110     DenseMatrix();
00111
00117     DenseMatrix(const DenseMatrix &in);
00118
00127     DenseMatrix(const int &num_rows, const int &num_cols);
00128
00154     DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00155
00189     DenseMatrix(const Real *const gen,
00190                 const int &gen_length,
00191                 const int &pro_length,
00192                 const bool &transpose);
00193
00195     ~DenseMatrix();
00196
00202     Matrix matrix_properties() const noexcept;
00203
00209     int num_rows() const noexcept;
00210
00216     int num_cols() const noexcept;
00217
00223     Real* data() const noexcept;
00224
00232     void SetOrdering(mtk::MatrixOrdering oo) noexcept;
00233
00242     Real GetValue(const int &row_coord, const int &col_coord) const noexcept;
00243
00251     void SetValue(const int &row_coord,
00252                  const int &col_coord,
00253                  const Real &val) noexcept;
00254
00256     void Transpose();
00257
00259     void OrderRowMajor();
00260
00262     void OrderColMajor();
00263
00274     static DenseMatrix Kron(const DenseMatrix &aa,
00275                             const DenseMatrix &bb);
00276
00286     bool WriteToFile(const std::string &filename) const;

```

```

00287
00288 private:
00289     Matrix matrix_properties;
00290
00291     Real *data_;
00292 };
00293 }
00294 #endif // End of: MTK_INCLUDE_MTK_DENSE_MATRIX_H_

```

18.27 include/mtk_div_1d.h File Reference

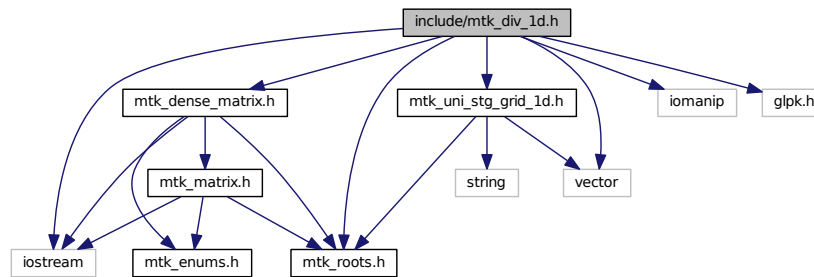
Includes the definition of the class Div1D.

```

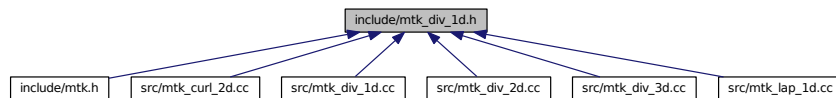
#include <iostream>
#include <iomanip>
#include <vector>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk_div_1d.h:



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



Classes

- class [mtk::Div1D](#)
Implements a 1D mimetic divergence operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.27.1 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](#).

18.28 mtk_div_1d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_DIV_1D_H_
00058 #define MTK_INCLUDE_DIV_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include <vector>
00064

```

```

00065 #include "glpk.h"
00066
00067 #include "mtk_roots.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_id.h"
00070
00071 namespace mtk {
00072
00083 class Div1D {
00084 public:
00086     friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00087
00089     Div1D();
00090
00096     Div1D(const Div1D &div);
00097
00099     ~Div1D();
00100
00106     bool ConstructDiv1D(int order_accuracy = kDefaultOrderAccuracy,
00107                        Real mimetic_threshold = kDefaultMimeticThreshold);
00108
00114     int num_bndy_coeffs() const;
00115
00121     Real *coeffs_interior() const;
00122
00128     Real *weights_crs(void) const;
00129
00135     Real *weights_cbs(void) const;
00136
00142     DenseMatrix mim_bndy() const;
00143
00149     std::vector<Real> sums_rows_mim_bndy() const;
00150
00156     DenseMatrix ReturnAsDenseMatrix(const
00157     UniStgGrid1D &grid) const;
00163     DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
00164     const;
00165 private:
00171     bool ComputeStencilInteriorGrid(void);
00172
00179     bool ComputeRationalBasisNullSpace(void);
00180
00186     bool ComputePreliminaryApproximations(void);
00187
00193     bool ComputeWeights(void);
00194
00200     bool ComputeStencilBoundaryGrid(void);
00201
00207     bool AssembleOperator(void);
00208
00209     int order_accuracy_;
00210     int dim_null_;
00211     int num_bndy_coeffs_;
00212     int divergence_length_;
00213     int minrow_;
00214     int row_;
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 *divergence_;
00224
00225     std::vector<Real> sums_rows_mim_bndy_;
00226
00227     Real mimetic_threshold_;
00228 };
00229 }
00230 #endif // End of: MTK_INCLUDE_DIV_1D_H_

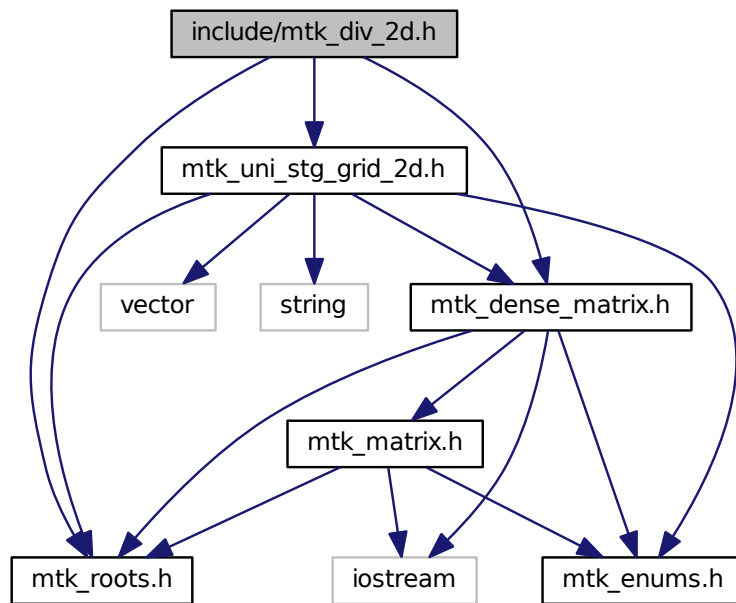
```

18.29 include/mtk_div_2d.h File Reference

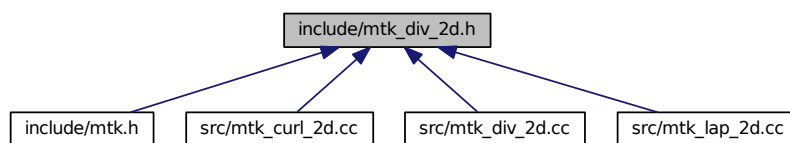
Includes the definition of the class Div2D.

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

Include dependency graph for mtk_div_2d.h:



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



Classes

- class [mtk::Div2D](#)

Implements a 2D mimetic divergence operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.29.1 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](#).

18.30 mtk_div_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_2D_H_
00058 #define MTK_INCLUDE_MTK_DIV_2D_H_
00059
00060 #include "mtk_roots.h"

```

```

00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Div2D {
00077 public:
00079     Div2D();
00080
00086     Div2D(const Div2D &div);
00087
00089     ~Div2D();
00090
00096     bool ConstructDiv2D(const UniStgGrid2D &grid,
00097                        int order_accuracy = kDefaultOrderAccuracy,
00098                        Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105     DenseMatrix ReturnAsDenseMatrix() const;
00106
00107 private:
00108     DenseMatrix divergence_;
00109
00110     int order_accuracy_;
00111
00112     Real mimetic_threshold_;
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_DIV_2D_H_

```

18.31 include/mtk_div_3d.h File Reference

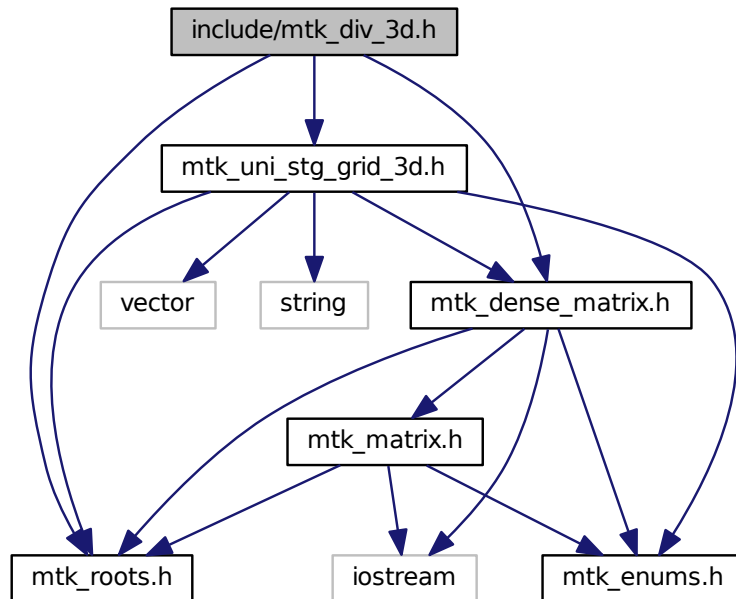
Includes the definition of the class Div3D.

```

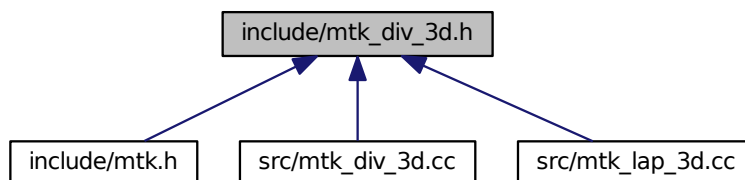
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for `mtk_div_3d.h`:



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



Classes

- class [mtk::Div3D](#)
Implements a 3D mimetic divergence operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.31.1 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](#).

18.32 mtk_div_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_DIV_3D_H_
00058 #define MTK_INCLUDE_MTK_DIV_3D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{

```

```

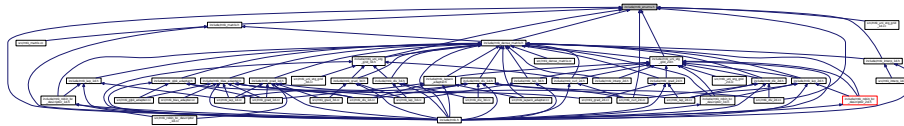
00065
00076 class Div3D {
00077     public:
00079         Div3D();
00080
00086         Div3D(const Div3D &div);
00087
00089         ~Div3D();
00090
00096         bool ConstructDiv3D(const UniStgGrid3D &grid,
00097                             int order_accuracy = kDefaultOrderAccuracy,
00098                             Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105         DenseMatrix ReturnAsDenseMatrix() const;
00106
00107     private:
00108         DenseMatrix divergence_;
00109
00110         int order_accuracy_;
00111
00112         Real mimetic_threshold_;
00113 };
00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_DIV_3D_H_

```

18.33 include/mtk_enums.h File Reference

Considered enumeration types in the MTK.

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



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Enumerations

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

Considered matrix storage schemes to implement sparse matrices.

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

Considered matrix ordering (for Fortran purposes).

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

Nature of the field discretized in a given grid.

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

Interpolation operator.

18.33.1 Detailed Description

Enumeration types are used throughout the MTK to differentiate instances of derived classes, as well as for mnemonic purposes. In this file, the enumeration types are listed alphabetically.

Author

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

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

18.34 mtk_enums.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_ENUMS_H_
00059 #define MTK_INCLUDE_ENUMS_H_
00060
00061 namespace mtk {
00062
00077 enum 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 }
00132 #endif // End of: MTK_INCLUDE_ENUMS_H_

```

18.35 include/mtk_glpk_adapter.h File Reference

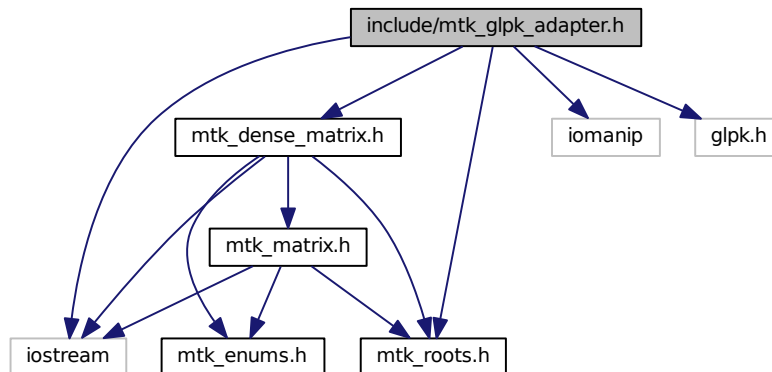
Adapter class for the GLPK API.

```

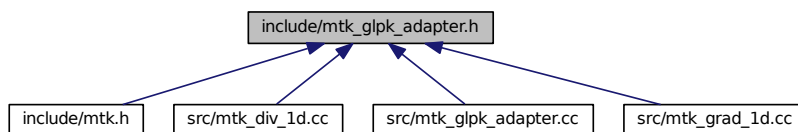
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"

```

Include dependency graph for mtk_glpk_adapter.h:



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



Classes

- class `mtk::GLPKAdapter`
Adapter class for the GLPK API.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

18.35.1 Detailed Description

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

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

See also

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

Author

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

Definition in file `mtk_glpk_adapter.h`.

18.36 mtk_glpk_adapter.h

```

00001
00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.

```

```

00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00067 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00068
00069 #include <iostream>
00070 #include <iomanip>
00071
00072 #include "glpk.h"
00073
00074 #include "mtk_roots.h"
00075 #include "mtk_dense_matrix.h"
00076
00077 namespace mtk {
00078
00102 class GLPKAdapter {
00103 public:
00124 static mtk::Real SolveSimplexAndCompare(
00125     mtk::Real *A,
00126     int nrows,
00127     int ncols,
00128     int kk,
00129     mtk::Real *hh,
00130     mtk::Real *qq,
00131     int robjective,
00132     mtk::Real mimetic_tol,
00133     int copy);
00134 };
00135 #endif // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_

```

18.37 include/mtk_grad_1d.h File Reference

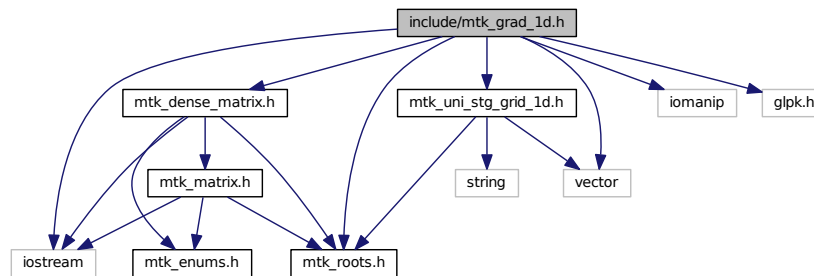
Includes the definition of the class Grad1D.

```

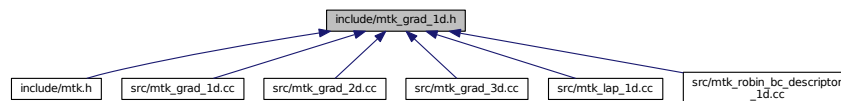
#include <iostream>
#include <iomanip>
#include <vector>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk_grad_1d.h:



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



Classes

- class [mtk::Grad1D](#)
Implements a 1D mimetic gradient operator.

Namespaces

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

18.37.1 Detailed Description

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

Author

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

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

18.38 mtk_grad_1d.h

00001

```

00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_GRAD_1D_H_
00058 #define MTK_INCLUDE_GRAD_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include <vector>
00064
00065 #include "glpk.h"
00066
00067 #include "mtk_roots.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_1d.h"
00070
00071 namespace mtk {
00072
00083 class Grad1D {
00084 public:
00085     friend std::ostream& operator <<(std::ostream& stream, Grad1D &in);
00086
00087     Grad1D();
00088
00089     Grad1D(const Grad1D &grad);
00090
00091     ~Grad1D();
00092
00093     bool ConstructGrad1D(int order_accuracy = kDefaultOrderAccuracy,
00094                         Real mimetic_threshold = kDefaultMimeticThreshold);
00095
00096     int num_bndy_coeffs() const;
00097
00098     Real *coeffs_interior() const;
00099
00100     Real *weights_crs(void) const;
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129

```



```

00135     Real *weights_cbs(void) const;
00136
00142     DenseMatrix mim_bndy() const;
00143
00149     std::vector<Real> sums_rows_mim_bndy() const;
00150
00156     DenseMatrix ReturnAsDenseMatrix(Real west,
Real east, int num_cells_x) const;
00157
00163     DenseMatrix ReturnAsDenseMatrix(const
UniStgGrid1D &grid) const;
00164
00170     DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
const;
00171
00172 private:
00178     bool ComputeStencilInteriorGrid(void);
00179
00186     bool ComputeRationalBasisNullSpace(void);
00187
00193     bool ComputePreliminaryApproximations(void);
00194
00200     bool ComputeWeights(void);
00201
00207     bool ComputeStencilBoundaryGrid(void);
00208
00214     bool AssembleOperator(void);
00215
00216     int order_accuracy_;
00217     int dim_null_;
00218     int num_bndy_approxs_;
00219     int num_bndy_coeffs_;
00220     int gradient_length_;
00221     int minrow_;
00222     int row_;
00223
00224     DenseMatrix rat_basis_null_space_;
00225
00226     Real *coeffs_interior_;
00227     Real *prem_apps_;
00228     Real *weights_crs_;
00229     Real *weights_cbs_;
00230     Real *mim_bndy_;
00231     Real *gradient_;
00232
00233     std::vector<Real> sums_rows_mim_bndy_;
00234
00235     Real mimetic_threshold_;
00236 };
00237 }
00238 #endif // End of: MTK_INCLUDE_GRAD_1D_H_

```

18.39 include/mtk_grad_2d.h File Reference

Includes the definition of the class Grad2D.

```

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

```


Mimetic Methods Toolkit namespace.

18.39.1 Detailed Description

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

Author

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

Definition in file [mtk_grad_2d.h](#).

18.40 mtk_grad_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_2D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_2D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{

```

```

00065
00076 class Grad2D {
00077 public:
00079     Grad2D();
00080
00086     Grad2D(const Grad2D &grad);
00087
00089     ~Grad2D();
00090
00096     bool ConstructGrad2D(const UniStgGrid2D &grid,
00097                         int order_accuracy = kDefaultOrderAccuracy,
00098                         Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105     DenseMatrix ReturnAsDenseMatrix() const;
00106
00107 private:
00108     DenseMatrix gradient_;
00109
00110     int order_accuracy_;
00111
00112     Real mimetic_threshold_;
00113 };
00114
00115 #endif // End of: MTK_INCLUDE_MTK_GRAD_2D_H_

```

18.41 include/mtk_grad_3d.h File Reference

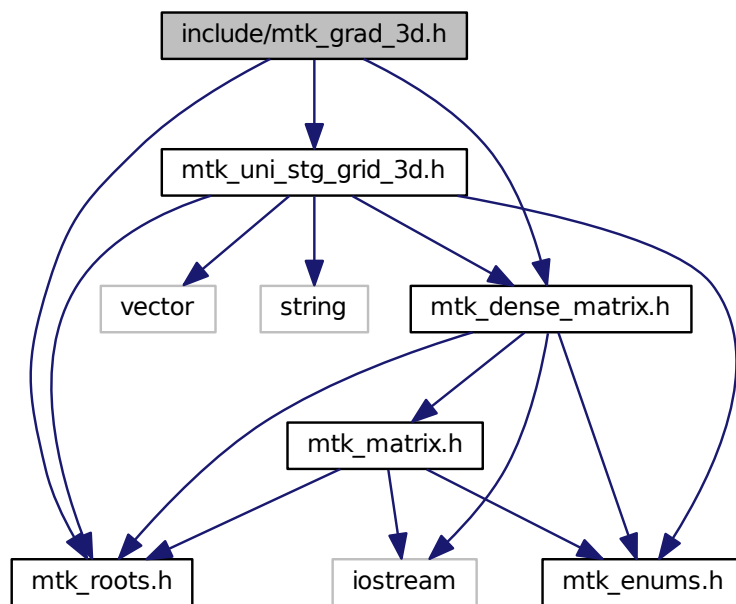
Includes the definition of the class Grad3D.

```

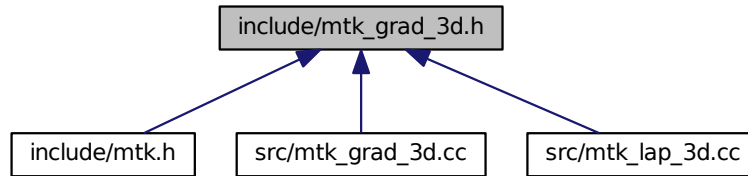
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_3d.h"

```

Include dependency graph for mtk_grad_3d.h:



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



Classes

- class `mtk::Grad3D`
Implements a 3D mimetic gradient operator.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

18.41.1 Detailed Description

This class implements a 3D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_3d.h](#).

18.42 mtk_grad_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026

```

```

00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_3D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_3D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{
00065
00066 class Grad3D {
00067 public:
00068   Grad3D();
00069
00070   Grad3D(const Grad3D &grad);
00071
00072   ~Grad3D();
00073
00074   bool ConstructGrad3D(const UniStgGrid3D &grid,
00075                       int order_accuracy = kDefaultOrderAccuracy,
00076                       Real mimetic_threshold = kDefaultMimeticThreshold);
00077
00078   DenseMatrix ReturnAsDenseMatrix() const;
00079
00080 private:
00081   DenseMatrix gradient_;
00082   int order_accuracy_;
00083   Real mimetic_threshold_;
00084 };
00085
00086 #endif // End of: MTK_INCLUDE_MTK_GRAD_3D_H_

```

18.43 include/mtk_interp_1d.h File Reference

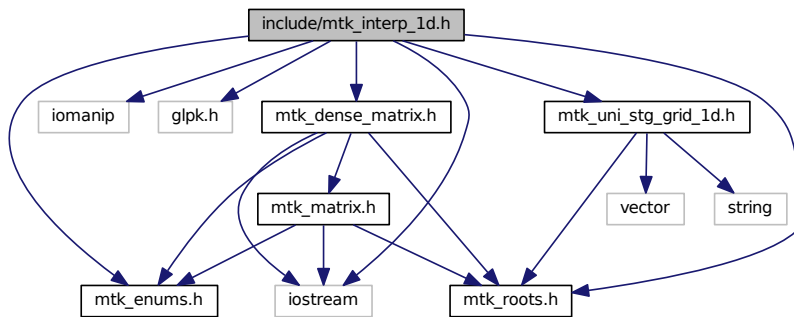
Includes the definition of the class Interp1D.

```

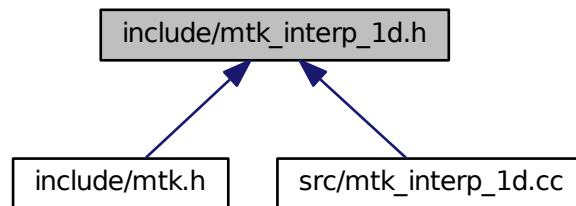
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk_interp_1d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::Interp1D`
Implements a 1D interpolation operator.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

18.43.1 Detailed Description

Definition of a class that implements a 1D interpolation operator.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu
 : Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file [mtk_interp_1d.h](#).

18.44 mtk_interp_1d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_INTERP_1D_H_
00059 #define MTK_INCLUDE_INTERP_1D_H_
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "glpk.h"
00065
00066 #include "mtk_roots.h"
00067 #include "mtk_enums.h"
00068 #include "mtk_dense_matrix.h"
00069 #include "mtk_uni_stg_grid_1d.h"
00070

```



```

00071 namespace mtk {
00072
00082 class Interp1D {
00083 public:
00085     friend std::ostream& operator <<(std::ostream& stream, Interp1D &in);
00086
00088     Interp1D();
00089
00095     Interp1D(const Interp1D &interp);
00096
00098     ~Interp1D();
00099
00105     bool ConstructInterp1D(int order_accuracy =
kDefaultOrderAccuracy,
00106                             mtk::DirInterp dir =
mtk::DirInterp::SCALAR_TO_VECTOR);
00107
00113     Real *coeffs_interior() const;
00114
00120     DenseMatrix ReturnAsDenseMatrix(const
UniStgGrid1D &grid) const;
00121
00122 private:
00123     DirInterp dir_interp_;
00124
00125     int order_accuracy_;
00126
00127     Real *coeffs_interior_;
00128 };
00129 }
00130 #endif // End of: MTK_INCLUDE_INTERP_1D_H_

```

18.45 include/mtk_interp_2d.h File Reference

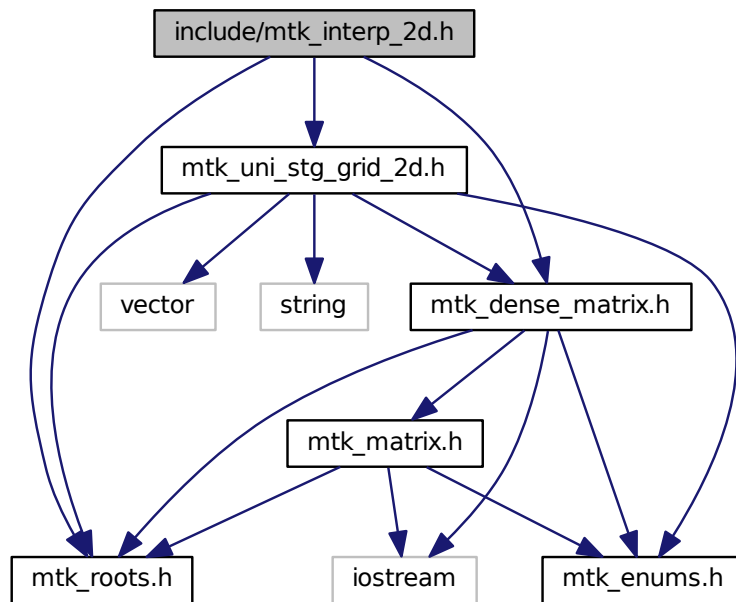
Includes the definition of the class Interp2D.

```

#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for `mtk_interp_2d.h`:



Classes

- class [mtk::Interp2D](#)
Implements a 2D interpolation operator.

Namespaces

- [mtk](#)
Mimetic Methods Toolkit namespace.

18.45.1 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](#).

18.46 mtk_interp_2d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_MTK_INTERP_2D_H_
00059 #define MTK_INCLUDE_MTK_INTERP_2D_H_
00060
00061 #include "mtk_roots.h"
00062 #include "mtk_dense_matrix.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk{
00066
00076 class Interp2D {
00077 public:
00079     Interp2D();
00080
00086     Interp2D(const Interp2D &interp);
00087
00089     ~Interp2D();
00090
00096     DenseMatrix ConstructInterp2D(const UniStgGrid2D &grid,
00097                                 int order_accuracy = kDefaultOrderAccuracy,
00098                                 Real mimetic_threshold =
00099                                 kDefaultMimeticThreshold);
00105     DenseMatrix ReturnAsDenseMatrix();
00106
00107 private:
00108     DenseMatrix interpolator_;
00109
00110     int order_accuracy_;
00111
00112     Real mimetic_threshold_;
00113 };

```

```

00114 }
00115 #endif // End of: MTK_INCLUDE_MTK_INTERP_2D_H_

```

18.47 include/mtk_lap_1d.h File Reference

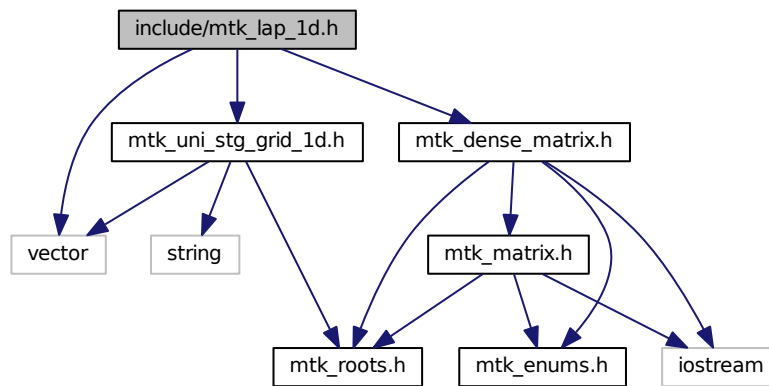
Includes the definition of the class Lap1D.

```

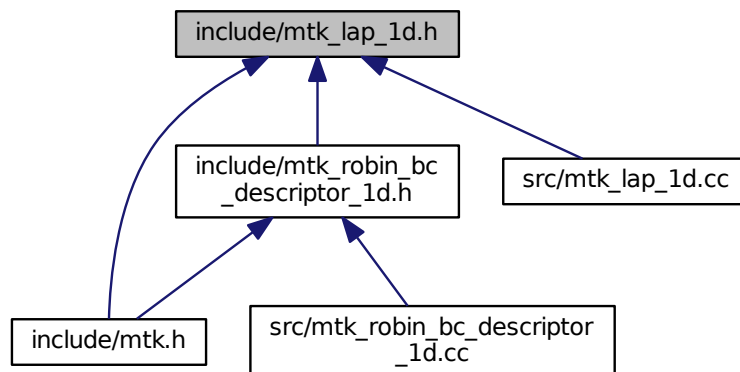
#include <vector>
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"

```

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.

18.47.1 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](#).

18.48 mtk_lap_1d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

```

```

00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_LAP_1D_H_
00058 #define MTK_INCLUDE_LAP_1D_H_
00059
00060 #include <vector>
00061
00062 #include "mtk_dense_matrix.h"
00063
00064 #include "mtk_uni_stg_grid_1d.h"
00065
00066 namespace mtk {
00067
00078 class Lap1D {
00079 public:
00081     friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00082
00084     Lap1D();
00085
00091     Lap1D(const Lap1D &lap);
00092
00094     ~Lap1D();
00095
00101     int order_accuracy() const;
00102
00108     Real mimetic_threshold() const;
00109
00115     Real delta() const;
00116
00122     bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00123                        Real mimetic_threshold = kDefaultMimeticThreshold);
00124
00130     std::vector<Real> sums_rows_mim_bndy() const;
00131
00137     DenseMatrix ReturnAsDenseMatrix(const
    UniStgGrid1D &grid) const;
00138
00144     const mtk::Real* data(const UniStgGrid1D &grid) const;
00145
00146 private:
00147     int order_accuracy_;
00148     int laplacian_length_;
00149
00150     Real *laplacian_;
00151
00152     mutable Real delta_;
00153
00154     Real mimetic_threshold_;
00155
00156     std::vector<Real> sums_rows_mim_bndy_;
00157 };
00158 }
00159 #endif // End of: MTK_INCLUDE_LAP_1D_H_

```

18.49 include/mtk_lap_2d.h File Reference

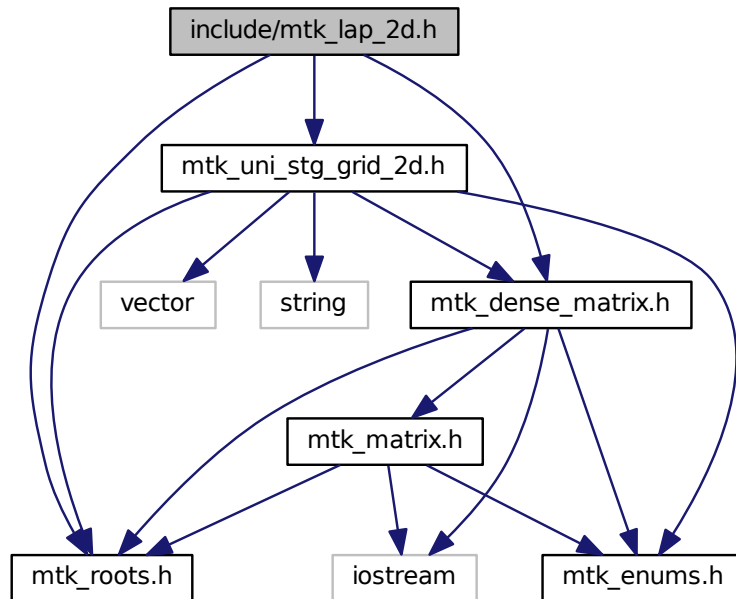
Includes the implementation of the class Lap2D.

```

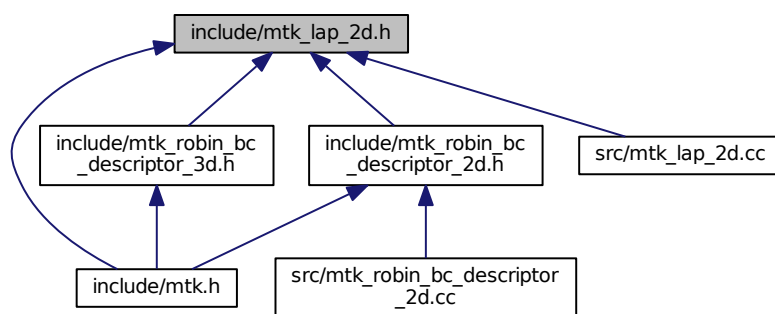
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk_lap_2d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::Lap2D`

Implements a 2D mimetic Laplacian operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.49.1 Detailed Description

This class implements a 2D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lap_2d.h](#).

18.50 mtk_lap_2d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_2D_H_
00058 #define MTK_INCLUDE_MTK_LAP_2D_H_
00059
00060 #include "mtk_roots.h"

```



```

00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap2D {
00077 public:
00078     Lap2D();
00080
00086     Lap2D(const Lap2D &lap);
00087
00089     ~Lap2D();
00090
00096     bool ConstructLap2D(const UniStgGrid2D &grid,
00097                        int order_accuracy = kDefaultOrderAccuracy,
00098                        Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105     DenseMatrix ReturnAsDenseMatrix() const;
00106
00112     Real *data() const;
00113
00114 private:
00115     DenseMatrix laplacian_;
00116
00117     int order_accuracy_;
00118
00119     Real mimetic_threshold_;
00120 };
00121 }
00122 #endif // End of: MTK_INCLUDE_MTK_LAP_2D_H_

```

18.51 include/mtk_lap_3d.h File Reference

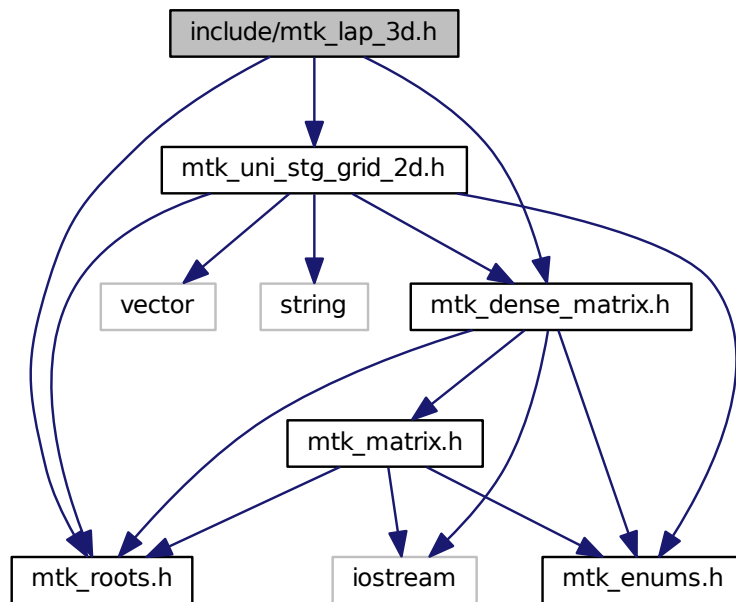
Includes the implementation of the class Lap3D.

```

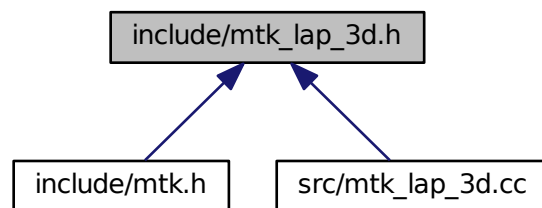
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for `mtk_lap_3d.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::Lap3D](#)

Implements a 3D mimetic Laplacian operator.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.51.1 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](#).

18.52 mtk_lap_3d.h

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_LAP_3D_H_
00058 #define MTK_INCLUDE_MTK_LAP_3D_H_
00059
00060 #include "mtk_roots.h"

```

```

00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_2d.h"
00063
00064 namespace mtk{
00065
00076 class Lap3D {
00077 public:
00079     UniStgGrid3D operator*(const UniStgGrid3D &grid) const;
00080
00082     Lap3D();
00083
00089     Lap3D(const Lap3D &lap);
00090
00092     ~Lap3D();
00093
00099     bool ConstructLap3D(const UniStgGrid3D &grid,
00100                        int order_accuracy = kDefaultOrderAccuracy,
00101                        Real mimetic_threshold = kDefaultMimeticThreshold);
00102
00108     DenseMatrix ReturnAsDenseMatrix() const;
00109
00115     Real *data() const;
00116
00117 private:
00118     DenseMatrix laplacian_;
00119
00120     int order_accuracy_;
00121
00122     Real mimetic_threshold_;
00123 };
00124
00125 #endif // End of: MTK_INCLUDE_MTK_LAP_3D_H_

```

18.53 include/mtk_lapack_adapter.h File Reference

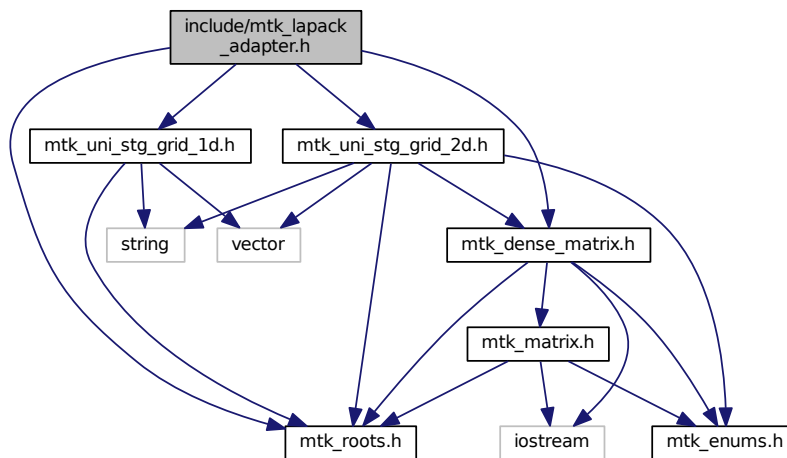
Adapter class for the LAPACK API.

```

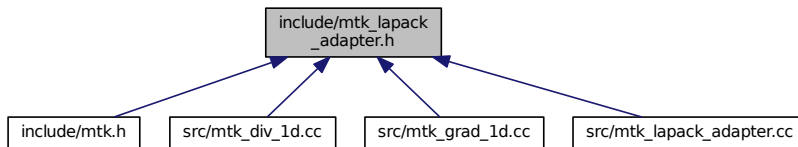
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for mtk_lapack_adapter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::LAPACKAdapter](#)
Adapter class for the LAPACK API.

Namespaces

- [mtk](#)
Mimetic Methods Toolkit namespace.

18.53.1 Detailed Description

Definition of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the LAPACK.

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lapack_adapter.h](#).

18.54 mtk_lapack_adapter.h

```

00001
00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026

```

```

00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065
00066 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00067 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00068
00069 #include "mtk_roots.h"
00070 #include "mtk_dense_matrix.h"
00071 #include "mtk_uni_stg_grid_ld.h"
00072 #include "mtk_uni_stg_grid_2d.h"
00073
00074 namespace mtk {
00075
00094 class LAPACKAdapter {
00095 public:
00106     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00107                                mtk::Real *rhs);
00108
00119     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00120                                mtk::DenseMatrix &rr);
00121
00132     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00133                                mtk::UniStgGrid1D &rhs);
00134
00146     static int SolveDenseSystem(mtk::DenseMatrix &mm,
00147                                mtk::UniStgGrid2D &rhs);
00148
00160     static int SolveRectangularDenseSystem(const
00161 mtk::DenseMatrix &aa,
00162                                           mtk::Real *ob_,
00163                                           int ob_ld_);
00175     static mtk::DenseMatrix QRFactorDenseMatrix(
00176 DenseMatrix &matrix);
00176 };
00177 }
00178 #endif // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_

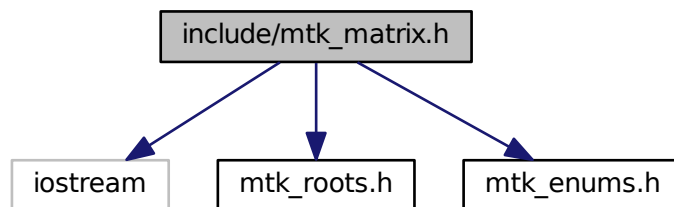
```

18.55 include/mtk_matrix.h File Reference

Definition of the representation of a matrix in the MTK.

```
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
```

Include dependency graph for mtk_matrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::Matrix](#)

Definition of the representation of a matrix in the MTK.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.55.1 Detailed Description

Definition of the representation for the matrices implemented in the MTK.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_matrix.h](#).

18.56 mtk_matrix.h

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_MATRIX_H_
00057 #define MTK_INCLUDE_MATRIX_H_
00058
00059 #include <iostream>
00060
00061 #include "mtk_roots.h"
00062 #include "mtk_enums.h"
00063
00064 namespace mtk {
00065
00075 class Matrix {
00076 public:
00077     Matrix();
00078
00079     Matrix(const Matrix &in);
00080
00081     ~Matrix() noexcept ;
00082
00083     MatrixStorage storage() const noexcept;
00084
00085     MatrixOrdering ordering() const noexcept;
00086
00087     int num_rows() const noexcept;
00088
00089     int num_cols() const noexcept;
00090
00091     int num_values() const noexcept;
00092
00093     int ld() const noexcept;
00094
00095     int num_zero() const noexcept;

```



```

00142
00148     int num_non_zero() const noexcept;
00149
00157     int num_null() const noexcept;
00158
00166     int num_non_null() const noexcept;
00167
00173     int kl() const noexcept;
00174
00180     int ku() const noexcept;
00181
00187     int bandwidth() const noexcept;
00188
00196     Real abs_density() const noexcept;
00197
00205     Real rel_density() const noexcept;
00206
00214     Real abs_sparsity() const noexcept;
00215
00223     Real rel_sparsity() const noexcept;
00224
00232     void set_storage(const MatrixStorage &tt) noexcept;
00233
00241     void set_ordering(const MatrixOrdering &oo) noexcept;
00242
00248     void set_num_rows(const int &num_rows) noexcept;
00249
00255     void set_num_cols(const int &num_cols) noexcept;
00256
00262     void set_num_zero(const int &in) noexcept;
00263
00269     void set_num_null(const int &in) noexcept;
00270
00272     void IncreaseNumZero() noexcept;
00273
00275     void IncreaseNumNull() noexcept;
00276
00277 private:
00278     MatrixStorage storage_;
00279
00280     MatrixOrdering ordering_;
00281
00282     int num_rows_;
00283     int num_cols_;
00284     int num_values_;
00285     int ld_;
00286
00287     int num_zero_;
00288     int num_non_zero_;
00289     int num_null_;
00290     int num_non_null_;
00291
00292     int kl_;
00293     int ku_;
00294     int bandwidth_;
00295
00296     Real abs_density_;
00297     Real rel_density_;
00298     Real abs_sparsity_;
00299     Real rel_sparsity_;
00300 };
00301 }
00302 #endif // End of: MTK_INCLUDE_MATRIX_H_

```

18.57 include/mtk_quad_1d.h File Reference

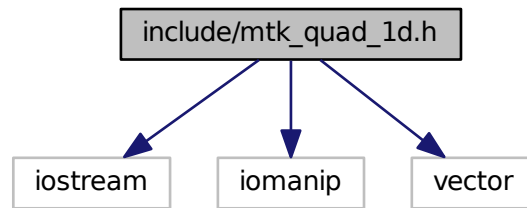
Includes the definition of the class Quad1D.

```

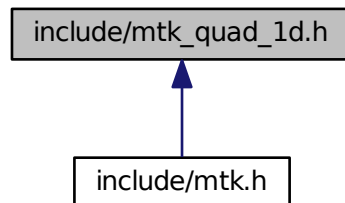
#include <iostream>
#include <iomanip>
#include <vector>

```

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.

18.57.1 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](#).

18.58 mtk_quad_1d.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_QUAD_1D_H_
00062 #define MTK_INCLUDE_QUAD_1D_H_
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #include <vector>
00068
00069 namespace mtk {
00070
00081 class Quad1D {
00082 public:
00084     friend std::ostream& operator <<(std::ostream& stream, Quad1D &in);
00085
00087     Quad1D();
00088
00094     Quad1D(const Quad1D &quad);
00095

```

```

00097 ~Quad1D();
00098
00104 int degree_approximation() const;
00105
00111 Real *weights() const;
00112
00121 Real Integrate(Real (*Integrand)(Real xx), UniStgGrid1D grid) const;
00122
00123 private:
00124     int degree_approximation_;
00125
00126     std::vector<Real> weights_;
00127 };
00128 }
00129 #endif // End of: MTK_INCLUDE_QUAD_1D_H_

```

18.59 include/mtk_robin_bc_descriptor_1d.h File Reference

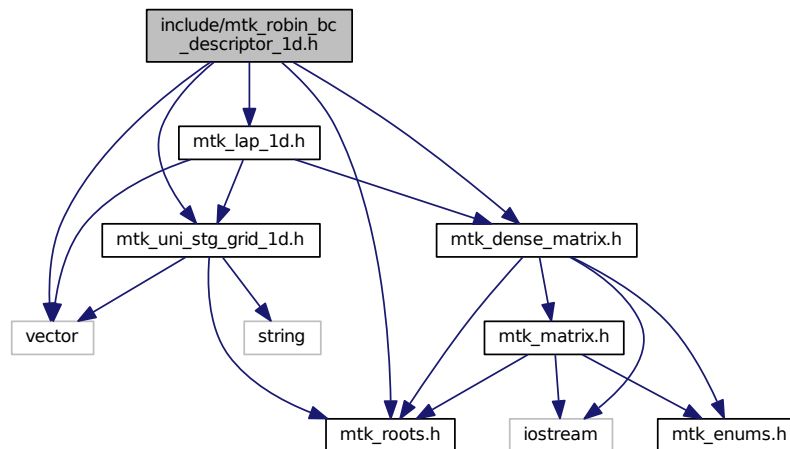
Impose Robin boundary conditions on the operators and on the grids.

```

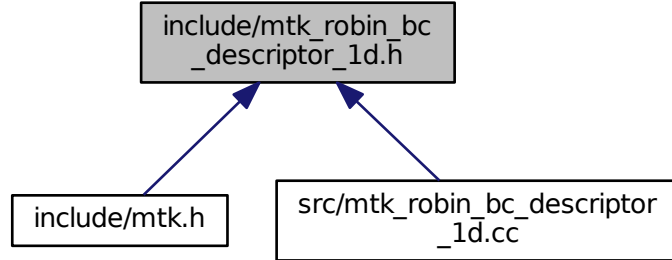
#include <vector>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_lap_1d.h"

```

Include dependency graph for mtk_robin_bc_descriptor_1d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::RobinBCDescriptor1D`
Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

Typedefs

- typedef `Real(* mtk::CoefficientFunction0D)(const Real &tt)`
A function of a BC coefficient evaluated on a 0D domain and time.

18.59.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 1D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b,t)u(b,t) + \eta_b(b,t)u'(b,t) = \beta_b(b,t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_robin_bc_descriptor_1d.h](#).

18.60 mtk_robin_bc_descriptor_1d.h

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include <vector>
00090
00091 #include "mtk_roots.h"
00092 #include "mtk_dense_matrix.h"
00093 #include "mtk_uni_stg_grid_1d.h"

```

```

00094 #include "mtk_lap_1d.h"
00095
00096 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00097 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_
00098
00099 namespace mtk {
00111 typedef Real (*CoefficientFunction0D)(const Real &tt);
00112
00155 class RobinBCDescriptor1D {
00156 public:
00158     RobinBCDescriptor1D();
00159
00165     RobinBCDescriptor1D(const RobinBCDescriptor1D &desc);
00166
00168     ~RobinBCDescriptor1D() noexcept;
00169
00175     int highest_order_diff_west() const noexcept;
00176
00182     int highest_order_diff_east() const noexcept;
00183
00189     void PushBackWestCoeff(CoefficientFunction0D cw);
00190
00196     void PushBackEastCoeff(CoefficientFunction0D ce);
00197
00203     void set_west_condition(Real (*west_condition)(const
Real &tt)) noexcept;
00204
00210     void set_east_condition(Real (*east_condition)(const
Real &tt)) noexcept;
00211
00221     bool ImposeOnLaplacianMatrix(const Lap1D &lap,
00222                                 DenseMatrix &matrix,
00223                                 const Real &time = mtk::kZero) const;
00230     void ImposeOnGrid(UniStgGrid1D &grid, const Real &time =
mtk::kZero) const;
00231
00232 private:
00233     int highest_order_diff_west_;
00234     int highest_order_diff_east_;
00235
00236     std::vector<CoefficientFunction0D> west_coefficients_;
00237     std::vector<CoefficientFunction0D> east_coefficients_;
00238
00239     Real (*west_condition_)(const Real &tt);
00240     Real (*east_condition_)(const Real &tt);
00241 };
00242
00243 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_1D_H_

```

18.61 include/mtk_robin_bc_descriptor_2d.h File Reference

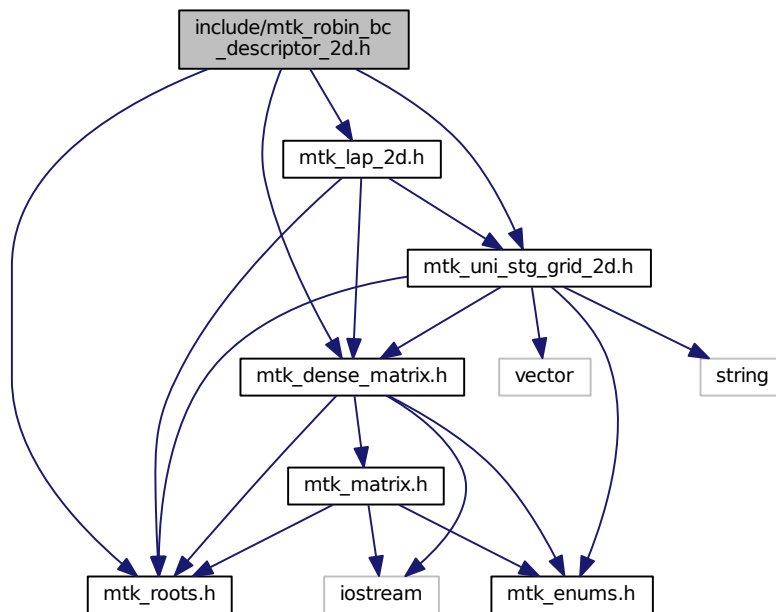
Impose Robin boundary conditions on the operators and on the grids.

```

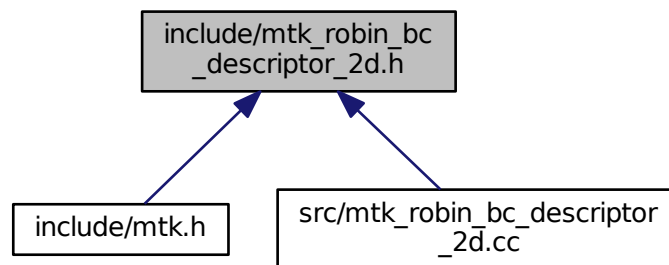
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for `mtk_robin_bc_descriptor_2d.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class [mtk::RobinBCDescriptor2D](#)

Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Typedefs

- typedef Real(* [mtk::CoefficientFunction1D](#))(const Real &xx, const Real &tt)

A function of a BC coefficient evaluated on a 1D domain and time.

18.61.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 2D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_robin_bc_descriptor_2d.h](#).

18.62 mtk_robin_bc_descriptor_2d.h

```
00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
```

```

00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_
00081 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_
00082
00083 #include "mtk_roots.h"
00084 #include "mtk_dense_matrix.h"
00085 #include "mtk_lap_2d.h"
00086 #include "mtk_uni_stg_grid_2d.h"
00087
00088 namespace mtk{
00089
00097 typedef Real (*CoefficientFunction1D) (const Real &xx, const
    Real &tt);
00098
00132 class RobinBCDescriptor2D {
00133 public:
00135     RobinBCDescriptor2D();
00136
00142     RobinBCDescriptor2D(const RobinBCDescriptor2D &desc);
00143
00145     ~RobinBCDescriptor2D() noexcept;
00146
00152     int highest_order_diff_west() const noexcept;
00153
00159     int highest_order_diff_east() const noexcept;
00160
00166     int highest_order_diff_south() const noexcept;
00167
00173     int highest_order_diff_north() const noexcept;
00174
00181     void PushBackWestCoeff(CoefficientFunction1D cw);
00182
00189     void PushBackEastCoeff(CoefficientFunction1D ce);
00190
00197     void PushBackSouthCoeff(CoefficientFunction1D cs);
00198
00205     void PushBackNorthCoeff(CoefficientFunction1D cn);
00206
00213     void set_west_condition(Real (*west_condition) (const
    Real &yy,
00214                                     const Real &tt)) noexcept;
00215
00222     void set_east_condition(Real (*east_condition) (const
    Real &yy,
00223                                     const Real &tt)) noexcept;
00224
00231     void set_south_condition(Real (*south_condition) (const
    Real &xx,
00232                                     const Real &tt)) noexcept;
00233
00240     void set_north_condition(Real (*north_condition) (const
    Real &xx,

```

```

00241                                     const Real &tt)) noexcept;
00242
00251     bool ImposeOnLaplacianMatrix(const Lap2D &lap,
00252                                   const UniStgGrid2D &grid,
00253                                   DenseMatrix &matrix,
00254                                   const Real &time = kZero) const;
00261     void ImposeOnGrid(UniStgGrid2D &grid, const Real &time =
00262                          kZero) const;
00263 private:
00272     bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00273                                         const UniStgGrid2D &grid,
00274                                         DenseMatrix &matrix,
00275                                         const Real &time = kZero) const;
00284     bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00285                                         const UniStgGrid2D &grid,
00286                                         DenseMatrix &matrix,
00287                                         const Real &time = kZero) const;
00296     bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00297                                         const UniStgGrid2D &grid,
00298                                         DenseMatrix &matrix,
00299                                         const Real &time = kZero) const;
00308     bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00309                                         const UniStgGrid2D &grid,
00310                                         DenseMatrix &matrix,
00311                                         const Real &time = kZero) const;
00320     bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00321                                         const UniStgGrid2D &grid,
00322                                         DenseMatrix &matrix,
00323                                         const Real &time = kZero) const;
00332     bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00333                                         const UniStgGrid2D &grid,
00334                                         DenseMatrix &matrix,
00335                                         const Real &time = kZero) const;
00344     bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00345                                         const UniStgGrid2D &grid,
00346                                         DenseMatrix &matrix,
00347                                         const Real &time = kZero) const;
00356     bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00357                                         const UniStgGrid2D &grid,
00358                                         DenseMatrix &matrix,
00359                                         const Real &time = kZero) const;
00360
00361     int highest_order_diff_west_;
00362     int highest_order_diff_east_;
00363     int highest_order_diff_south_;
00364     int highest_order_diff_north_;
00365
00366     std::vector<CoefficientFunction1D> west_coefficients_;
00367     std::vector<CoefficientFunction1D> east_coefficients_;
00368     std::vector<CoefficientFunction1D> south_coefficients_;
00369     std::vector<CoefficientFunction1D> north_coefficients_;
00370
00371     Real (*west_condition_)(const Real &xx, const Real &tt);
00372     Real (*east_condition_)(const Real &xx, const Real &tt);
00373     Real (*south_condition_)(const Real &yy, const Real &tt);
00374     Real (*north_condition_)(const Real &yy, const Real &tt);
00375 };
00376 }
00377 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_2D_H_

```

18.63 include/mtk_robin_bc_descriptor_3d.h File Reference

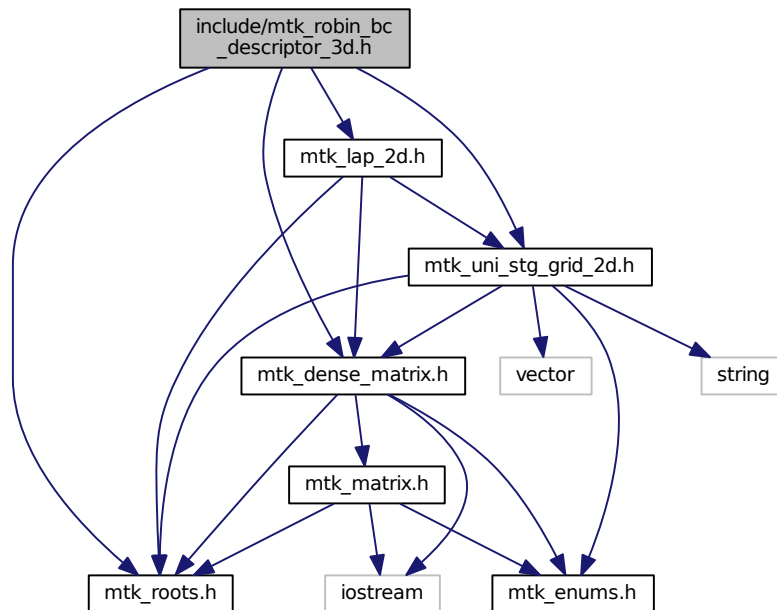
Impose Robin boundary conditions on the operators and on the grids.

```

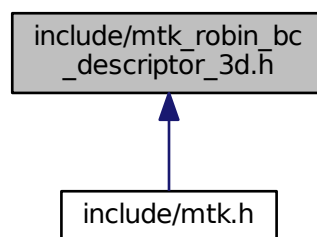
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_lap_2d.h"
#include "mtk_uni_stg_grid_2d.h"

```

Include dependency graph for `mtk_robin_bc_descriptor_3d.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::RobinBCDescriptor3D`

Impose Robin boundary conditions on the operators and on the grids.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Typedefs

- typedef Real(* [mtk::CoefficientFunction2D](#))(const Real &xx, const Real &yy, const Real &tt)

A function of a BC coefficient evaluated on a 2D domain and time.

18.63.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 3D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary. These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_robin_bc_descriptor_3d.h](#).

18.64 mtk_robin_bc_descriptor_3d.h

```
00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
```

```

00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #ifndef MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00081 #define MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_
00082
00083 #include "mtk_roots.h"
00084 #include "mtk_dense_matrix.h"
00085 #include "mtk_lap_2d.h"
00086 #include "mtk_uni_stg_grid_2d.h"
00087
00088 namespace mtk{
00089
00097 typedef Real (*CoefficientFunction2D) (const Real &xx,
00098                                       const Real &yy,
00099                                       const Real &tt);
00100
00134 class RobinBCDescriptor3D {
00135 public:
00137   RobinBCDescriptor3D();
00138
00144   RobinBCDescriptor3D(const RobinBCDescriptor3D &desc);
00145
00147   ~RobinBCDescriptor3D() noexcept;
00148
00154   int highest_order_diff_west() const noexcept;
00155
00156   // ...
00157
00164   void PushBackWestCoeff(CoefficientFunction2D cw);
00165
00166   // ...
00167
00174   void set_west_condition(Real (*west_condition) (const
Real &xx,
00175                                                         const Real &yy,
00176                                                         const Real &tt)) noexcept;
00177
00178   // ...
00179
00188   bool ImposeOnLaplacianMatrix(const Lap3D &lap,
00189                                const UniStgGrid3D &grid,
00190                                DenseMatrix &matrix,
00191                                const Real &time = kZero) const;
00198   void ImposeOnGrid(UniStgGrid3D &grid, const Real &time =
kZero) const;
00199
00200 private:
00209   bool ImposeOnSouthBoundaryNoSpace(const Lap2D &lap,
00210                                     const UniStgGrid2D &grid,
00211                                     DenseMatrix &matrix,
00212                                     const Real &time = kZero) const;
00221   bool ImposeOnNorthBoundaryNoSpace(const Lap2D &lap,
00222                                     const UniStgGrid2D &grid,

```

```

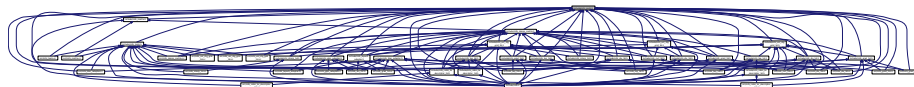
00223         DenseMatrix &matrix,
00224         const Real &time = kZero) const;
00233 bool ImposeOnWestBoundaryNoSpace(const Lap2D &lap,
00234         const UniStgGrid2D &grid,
00235         DenseMatrix &matrix,
00236         const Real &time = kZero) const;
00245 bool ImposeOnEastBoundaryNoSpace(const Lap2D &lap,
00246         const UniStgGrid2D &grid,
00247         DenseMatrix &matrix,
00248         const Real &time = kZero) const;
00257 bool ImposeOnSouthBoundaryWithSpace(const Lap2D &lap,
00258         const UniStgGrid2D &grid,
00259         DenseMatrix &matrix,
00260         const Real &time = kZero) const;
00269 bool ImposeOnNorthBoundaryWithSpace(const Lap2D &lap,
00270         const UniStgGrid2D &grid,
00271         DenseMatrix &matrix,
00272         const Real &time = kZero) const;
00281 bool ImposeOnWestBoundaryWithSpace(const Lap2D &lap,
00282         const UniStgGrid2D &grid,
00283         DenseMatrix &matrix,
00284         const Real &time = kZero) const;
00293 bool ImposeOnEastBoundaryWithSpace(const Lap2D &lap,
00294         const UniStgGrid2D &grid,
00295         DenseMatrix &matrix,
00296         const Real &time = kZero) const;
00297
00298 int highest_order_diff_west_;
00299 int highest_order_diff_east_;
00300 int highest_order_diff_south_;
00301 int highest_order_diff_north_;
00302 int highest_order_diff_bottom_;
00303 int highest_order_diff_top_;
00304
00305 std::vector<CoefficientFunction2D> west_coefficients_;
00306 std::vector<CoefficientFunction2D> east_coefficients_;
00307 std::vector<CoefficientFunction2D> south_coefficients_;
00308 std::vector<CoefficientFunction2D> north_coefficients_;
00309 std::vector<CoefficientFunction2D> bottom_coefficients_;
00310 std::vector<CoefficientFunction2D> top_coefficients_;
00311
00312 Real (*west_condition_)(const Real &xx,
00313         const Real &yy,
00314         const Real &tt);
00315 Real (*east_condition_)(const Real &xx,
00316         const Real &yy,
00317         const Real &tt);
00318 Real (*south_condition_)(const Real &xx,
00319         const Real &yy,
00320         const Real &tt);
00321 Real (*north_condition_)(const Real &xx,
00322         const Real &yy,
00323         const Real &tt);
00324 Real (*bottom_condition_)(const Real &xx,
00325         const Real &yy,
00326         const Real &tt);
00327 Real (*top_condition_)(const Real &xx,
00328         const Real &yy,
00329         const Real &tt);
00330 };
00331 }
00332 #endif // End of: MTK_INCLUDE_ROBIN_BC_DESCRIPTOR_3D_H_

```

18.65 include/mtk_roots.h File Reference

Fundamental definitions to be used across all classes of the MTK.

This graph shows which files directly or indirectly include this file:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Typedefs

- typedef float [mtk::Real](#)

Users can simply change this to build a double- or single-precision MTK.

Variables

- const float [mtk::kZero](#) {0.0f}

MTK's zero defined according to selective compilation.

- const float [mtk::kOne](#) {1.0f}

MTK's one defined according to selective compilation.

- const float [mtk::kTwo](#) {2.0f}

MTK's two defined according to selective compilation.

- const float [mtk::kDefaultTolerance](#) {1e-7f}

Considered tolerance for comparisons in numerical methods.

- const float [mtk::kDefaultMimeticThreshold](#) {1e-6f}

Default tolerance for higher-order mimetic operators.

- const int [mtk::kDefaultOrderAccuracy](#) {2}

Default order of accuracy for mimetic operators.

- const int [mtk::kCriticalOrderAccuracyGrad](#) {10}

At this order (and higher) we must use the CBSA to construct gradients.

- const int [mtk::kCriticalOrderAccuracyDiv](#) {8}

At this order (and higher) we must use the CBSA to construct divergences.

18.65.1 Detailed Description

This file contains the fundamental definitions that classes of the MTK rely on to be implemented. Examples of these definitions are the definition of fundamental data types, and global variables affecting the construction of mimetic operators, among others.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at sciences dot sdsu dot edu

Todo Test selective precision mechanisms.

Definition in file [mtk_roots.h](#).

18.66 mtk_roots.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_ROOTS_H_
00062 #define MTK_INCLUDE_ROOTS_H_
00063
00064 namespace mtk {
00070
00090 #ifdef MTK_PRECISION_DOUBLE
00091 typedef double Real;
00092 #else
00093 typedef float Real;
00094 #endif
00095
00121 #ifdef MTK_PRECISION_DOUBLE
00122 const double kZero{0.0};
00123 const double kOne{1.0};
00124 const double kTwo{2.0};
00125 #else
00126 const float kZero{0.0f};
00127 const float kOne{1.0f};
00128 const float kTwo{2.0f};
00129 #endif
00130
00140 #ifdef MTK_PRECISION_DOUBLE
00141 const double kDefaultTolerance{1e-7};
00142 #else
00143 const float kDefaultTolerance{1e-7f};
00144 #endif
00145
00155 #ifdef MTK_PRECISION_DOUBLE
00156 const double kDefaultMimeticThreshold{1e-6};
00157 #else
00158 const float kDefaultMimeticThreshold{1e-6f};

```


Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Note

Performance Tip 8.1. If they do not need to be modified by the called function, pass large objects using pointers to constant data or references to constant data, to obtain the performance benefits of pass-by-reference.

Definition in file [mtk_tools.h](#).

18.68 mtk_tools.h

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_TOOLS_H_
00062 #define MTK_INCLUDE_TOOLS_H_
00063
00064 #include <ctime>
00065
00066 #include "mtk_roots.h"
00067
00068 namespace mtk {
00069
00080 class Tools {
00081 public:
00092     static void Prevent(const bool complement,
00093                        const char *const fname,

```

```

00094         int lineno,
00095         const char *const fxname) noexcept;
00096
00102     static void BeginUnitTestNo(const int &nn) noexcept;
00103
00109     static void EndUnitTestNo(const int &nn) noexcept;
00110
00116     static void Assert(const bool &condition) noexcept;
00117
00118 private:
00119     static int test_number_;
00120
00121     static Real duration_;
00122
00123     static clock_t begin_time_;
00124 };
00125 }
00126 #endif // End of: MTK_INCLUDE_TOOLS_H_

```

18.69 include/mtk_uni_stg_grid_1d.h File Reference

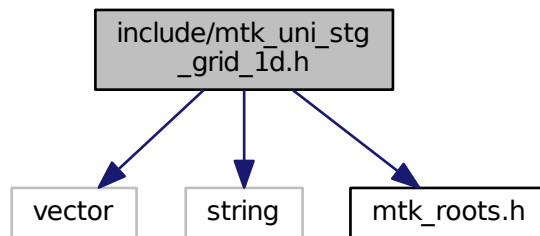
Definition of an 1D uniform staggered grid.

```

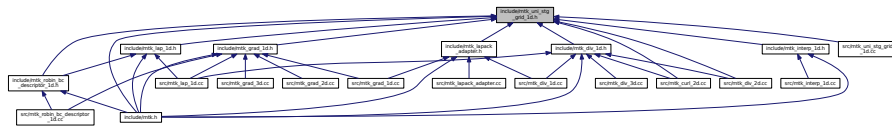
#include <vector>
#include <string>
#include "mtk_roots.h"

```

Include dependency graph for mtk_uni_stg_grid_1d.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::UniStgGrid1D`
Uniform 1D Staggered Grid.

Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

18.69.1 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](#).

18.70 mtk_uni_stg_grid_1d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_

```

```

00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078 public:
00080     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
00083     UniStgGrid1D();
00084
00090     UniStgGrid1D(const UniStgGrid1D &grid);
00091
00102     UniStgGrid1D(const Real &west_bndy_x,
00103                  const Real &east_bndy_x,
00104                  const int &num_cells_x,
00105                  const mtk::FieldNature &nature =
00106 mtk::FieldNature::SCALAR);
00106
00108     ~UniStgGrid1D();
00109
00115     Real west_bndy_x() const;
00116
00122     Real east_bndy_x() const;
00123
00129     Real delta_x() const;
00130
00138     const Real *discrete_domain_x() const;
00139
00147     Real *discrete_field();
00148
00154     int num_cells_x() const;
00155
00161     void BindScalarField(Real (*ScalarField)(const Real &xx));
00162
00173     void BindVectorField(Real (*VectorField)(Real xx));
00174
00186     bool WriteToFile(std::string filename,
00187                     std::string space_name,
00188                     std::string field_name) const;
00189
00190 private:
00191     FieldNature nature_;
00192
00193     std::vector<Real> discrete_domain_x_;
00194     std::vector<Real> discrete_field_;
00195
00196     Real west_bndy_x_;
00197     Real east_bndy_x_;
00198     Real num_cells_x_;
00199     Real delta_x_;
00200 };
00201 }
00202 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_

```

18.71 include/mtk_uni_stg_grid_2d.h File Reference

Definition of an 2D uniform staggered grid.

```

#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"

```

```
graph TD; Root["include/mtk_uni_stg_grid_2d.h"] --> vector; Root --> string; Root --> mtk_dense_matrix["mtk_dense_matrix.h"]; Root --> mtk_roots["mtk_roots.h"]; Root --> iostream; Root --> mtk_enums["mtk_enums.h"]; mtk_dense_matrix --> mtk_roots; mtk_dense_matrix --> iostream; mtk_dense_matrix --> mtk_enums; mtk_matrix["mtk_matrix.h"] --> iostream; mtk_matrix --> mtk_enums;
```

- class `mtk::UniStgGrid2D`
Uniform 2D Staggered Grid.

- `mtk`
Mimetic Methods Toolkit namespace.

Definition of an 2D uniform staggered grid.

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_2d.h](#).

18.72 mtk_uni_stg_grid_2d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_2D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_2D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {
00069
00070 class UniStgGrid2D {
00071 public:
00072     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid2D &in);
00073
00074     UniStgGrid2D();
00075
00076     UniStgGrid2D(const UniStgGrid2D &grid);
00077
00078     UniStgGrid2D(const Real &west_bndy_x,
00079                 const Real &east_bndy_x,
00080                 const int &num_cells_x,
00081                 const Real &south_bndy_y,
00082                 const Real &north_bndy_y,
00083                 const int &num_cells_y,
00084                 const mtk::FieldNature &nature =
00085 mtk::FieldNature::SCALAR);
00086
00087     ~UniStgGrid2D();
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117

```



```

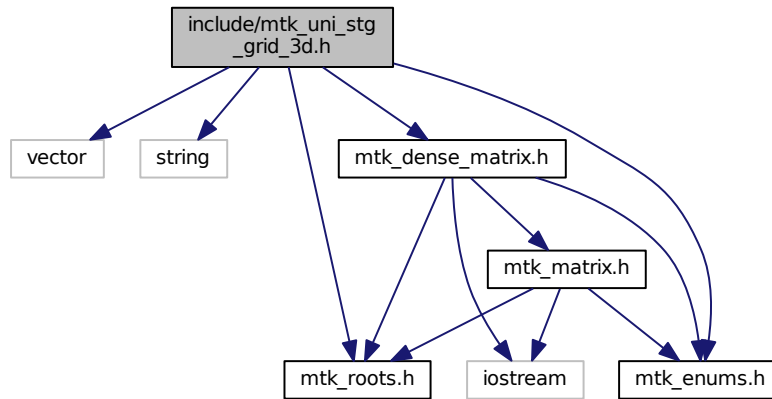
00125     const Real *discrete_domain_x() const;
00126
00134     const Real *discrete_domain_y() const;
00135
00141     Real *discrete_field();
00142
00150     FieldNature nature() const;
00151
00157     Real west_bndy() const;
00158
00164     Real east_bndy() const;
00165
00171     int num_cells_x() const;
00172
00178     Real delta_x() const;
00179
00185     Real south_bndy() const;
00186
00192     Real north_bndy() const;
00193
00199     int num_cells_y() const;
00200
00206     Real delta_y() const;
00207
00213     bool Bound() const;
00214
00220     int Size() const;
00221
00227     void BindScalarField(Real (*ScalarField)(const Real &xx, const
Real &yy));
00228
00242     void BindVectorField(Real (*VectorFieldPComponent)(const
Real &xx,
00243                                     const Real &yy),
00244                             Real (*VectorFieldQComponent)(const Real &xx,
00245                                     const Real &yy));
00246
00259     bool WriteToFile(std::string filename,
00260                     std::string space_name_x,
00261                     std::string space_name_y,
00262                     std::string field_name) const;
00263
00264 private:
00276     void BindVectorFieldPComponent(
00277         Real (*VectorFieldPComponent)(const Real &xx, const Real &yy));
00278
00290     void BindVectorFieldQComponent(
00291         Real (*VectorFieldQComponent)(const Real &xx, const Real &yy));
00292
00293     std::vector<Real> discrete_domain_x_;
00294     std::vector<Real> discrete_domain_y_;
00295     std::vector<Real> discrete_field_;
00296
00297     FieldNature nature_;
00298
00299     Real west_bndy_;
00300     Real east_bndy_;
00301     int num_cells_x_;
00302     Real delta_x_;
00303
00304     Real south_bndy_;
00305     Real north_bndy_;
00306     int num_cells_y_;
00307     Real delta_y_;
00308 };
00309 }
00310 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_

```

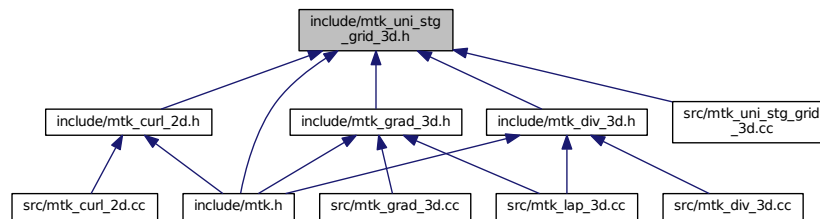
18.73 include/mtk_uni_stg_grid_3d.h File Reference

Definition of an 3D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
Include dependency graph for mtk_uni_stg_grid_3d.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `mtk::UniStgGrid3D`
Uniform 3D Staggered Grid.

Namespaces

- `mtk`
Mimetic Methods Toolkit namespace.

18.73.1 Detailed Description

Definition of an 3D uniform staggered grid.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Todo Create overloaded binding routines that read data from files.

Definition in file [mtk_uni_stg_grid_3d.h](#).

18.74 mtk_uni_stg_grid_3d.h

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_3D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_3D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {

```

```

00069
00079 class UniStgGrid3D {
00080 public:
00082     friend std::ostream& operator <<(std::ostream& stream, UniStgGrid3D &in);
00083
00091     UniStgGrid3D operator=(const UniStgGrid3D &in);
00092
00094     UniStgGrid3D();
00095
00101     UniStgGrid3D(const UniStgGrid3D &grid);
00102
00119     UniStgGrid3D(const Real &west_bndy_x,
00120                  const Real &east_bndy_x,
00121                  const int &num_cells_x,
00122                  const Real &south_bndy_y,
00123                  const Real &north_bndy_y,
00124                  const int &num_cells_y,
00125                  const Real &bottom_bndy_z,
00126                  const Real &top_bndy_z,
00127                  const int &num_cells_z,
00128                  const mtk::FieldNature &nature =
mtk::FieldNature::SCALAR);
00129
00131     ~UniStgGrid3D();
00132
00140     const Real *discrete_domain_x() const;
00141
00149     const Real *discrete_domain_y() const;
00150
00158     const Real *discrete_domain_z() const;
00159
00165     Real *discrete_field();
00166
00174     FieldNature nature() const;
00175
00181     Real west_bndy() const;
00182
00188     Real east_bndy() const;
00189
00195     int num_cells_x() const;
00196
00202     Real delta_x() const;
00203
00209     Real south_bndy() const;
00210
00216     Real north_bndy() const;
00217
00223     int num_cells_y() const;
00224
00230     Real delta_y() const;
00231
00237     Real bottom_bndy() const;
00238
00244     Real top_bndy() const;
00245
00251     int num_cells_z() const;
00252
00258     Real delta_z() const;
00259
00265     bool Bound() const;
00266
00272     int Size() const;
00273
00279     void BindScalarField(
00280         Real (*ScalarField)(const Real &xx, const Real &yy, const Real &zz));
00281
00298     void BindVectorField(Real (*VectorFieldPComponent)(const
Real &xx,
00299                                     const Real &yy,
00300                                     const Real &zz),
00301                          Real (*VectorFieldQComponent)(const Real &xx,
00302                                                         const Real &yy,
00303                                                         const Real &zz),
00304                          Real (*VectorFieldRComponent)(const Real &xx,
00305                                                         const Real &yy,
00306                                                         const Real &zz));
00307
00321     bool WriteToFile(std::string filename,
00322                     std::string space_name_x,
00323                     std::string space_name_y,
00324                     std::string space_name_z,

```

```

00325         std::string field_name) const;
00326
00327 private:
00340 void BindVectorFieldPComponent (
00341     Real (*VectorFieldPComponent) (const Real &xx,
00342                                     const Real &yy,
00343                                     const Real &zz));
00344
00357 void BindVectorFieldQComponent (
00358     Real (*VectorFieldQComponent) (const Real &xx,
00359                                     const Real &yy,
00360                                     const Real &zz));
00361
00374 void BindVectorFieldRComponent (
00375     Real (*VectorFieldRComponent) (const Real &xx,
00376                                     const Real &yy,
00377                                     const Real &zz));
00378
00379 std::vector<Real> discrete_domain_x_;
00380 std::vector<Real> discrete_domain_y_;
00381 std::vector<Real> discrete_domain_z_;
00382 std::vector<Real> discrete_field_;
00383
00384 FieldNature nature_;
00385
00386 Real west_bndy_;
00387 Real east_bndy_;
00388 int num_cells_x_;
00389 Real delta_x_;
00390
00391 Real south_bndy_;
00392 Real north_bndy_;
00393 int num_cells_y_;
00394 Real delta_y_;
00395
00396 Real bottom_bndy_;
00397 Real top_bndy_;
00398 int num_cells_z_;
00399 Real delta_z_;
00400 };
00401 }
00402 #endif // End of: MTK_INCLUDE_UNI_STG_GRID_3D_H_

```

18.75 Makefile.inc File Reference

18.76 Makefile.inc

```

00001 # Makefile setup file for the MTK.
00002
00003 SHELL := /bin/bash
00004
00005 # 1. Absolute path to base directory of the MTK.
00006 # _____
00007
00008 BASE = /home/esanchez/Dropbox/MTK
00009
00010 # 2. The machine (platform) identifier and required machine precision.
00011 # _____
00012
00013 # Options are:
00014 # - LINUX: A LINUX box installation.
00015 # - OSX: Uses OS X optimized solvers.
00016
00017 PLAT = LINUX
00018
00019 # Options are:
00020 # - SINGLE: Use 4 B floating point numbers.
00021 # - DOUBLE: Use 8 B floating point numbers.
00022
00023 PRECISION = DOUBLE
00024
00025 # 3. Optimized solvers and operations by means of ATLAS in Linux?
00026 # _____
00027
00028 # If you have selected OSX in step 1, then you don't need to worry about this.

```

```

00029
00030 # Options are ON xor OFF:
00031
00032 ATL_OPT = OFF
00033
00034 # 4. Paths to dependencies (header files for compiling).
00035 # _____
00036
00037 # GLPK include path (soon to go):
00038
00039 GLPK_INC = $(HOME)/Libraries/glpk-4.35/include
00040
00041 # Linux: If ATLAS optimization is ON, users should only provide the path to
00042 # ATLAS:
00043
00044 ATLAS_INC = $(HOME)/Libraries/ATLAS_3.8.4-CORE/include
00045
00046 # OS X: Do nothing.
00047
00048 # 5. Paths to dependencies (archive files for (static) linking).
00049 # _____
00050
00051 # GLPK linking path (soon to go):
00052
00053 GLPK_LIB = $(HOME)/Libraries/glpk-4.35/lib/lib64/libglpk.a
00054
00055 # If optimization is OFF, then provide the paths for:
00056
00057 BLAS_LIB = $(HOME)/Libraries/BLAS-3.5.0/libblas.a
00058 LAPACK_LIB = $(HOME)/Libraries/lapack-3.5.0/liblapack.a
00059
00060 # WARNING: Vendor libraries should be used whenever they are available.
00061
00062 # However, if optimization is ON, please provide the path the ATLAS' archive:
00063
00064 ATLAS_LIB = $(HOME)/Libraries/ATLAS_3.8.4-CORE/ATLAS_3.8.4-BUILD-Citadel/lib
00065
00066 # 6. Compiler and its flags.
00067 # _____
00068
00069 CC = g++
00070
00071 # Selective Verbose Execution for Quick Debugging. Options are defined per
00072 # concern, and per data hierarchy on each concern.
00073
00074 # 0: NO verbose at all.
00075
00076 # 1: Enable verbose down to the 7th concern: messages.
00077 # 2: Enable verbose down to the 7th concern: messages + scalar results.
00078 # 3: Enable verbose down to the 7th concern. 1.1. + array results.
00079 # 4: Enable verbose down to the 7th concern. 1.2. + matrix results.
00080
00081 # 5: Enable verbose down to the 6th concern: messages.
00082 # 6: Enable verbose down to the 6th concern: messages + scalar results.
00083 # 7: Enable verbose down to the 6th concern. 2.1. + array results.
00084 # 8: Enable verbose down to the 6th concern. 2.2. + matrix results.
00085
00086 # 9: Enable verbose down to the 5th concern: messages.
00087 # 10: Enable verbose down to the 5th concern: messages + scalar results.
00088 # 11: Enable verbose down to the 5th concern. 3.1. + array results.
00089 # 12: Enable verbose down to the 5th concern. 3.2. + matrix results.
00090
00091 # 13: Enable verbose down to the 4th concern: messages.
00092 # 14: Enable verbose down to the 4th concern: messages + scalar results.
00093 # 15: Enable verbose down to the 4th concern. 4.1. + array results.
00094 # 16: Enable verbose down to the 4th concern. 4.2. + matrix results.
00095
00096 VERBOSE_LEVEL = 16
00097
00098 # Enable preventions. In the MTK, methods first validate their required
00099 # pre-conditions in run-time. Similarly, in many points throughout the MTK
00100 # codebase, different sanity checks are performed, as well. If this symbol is
00101 # defined to be 0, the MTK will # perform no validations to enhance execution
00102 # performance. Options are:
00103 # - YES.
00104 # - NO.
00105
00106 PERFORM_PREVENTIONS = YES
00107
00108 # Enables creation of LaTeX tables verbosing the computation of mimetic weights.
00109

```

```

00110 VERBOSE_WEIGHTS = YES
00111
00112 # Flags recommended for release code:
00113
00114 CCFLAGS = -Wall -Werror -O2
00115
00116 # Flags recommended for debugging code:
00117
00118 CCFLAGS = -Wall -Werror -g
00119
00120 # 7. Archiver, its flags, and ranlib:
00121 #
00122
00123 ARCH = ar
00124 ARCHFLAGS = cr
00125
00126 # If your system does not have "ranlib" then set: "RANLIB = echo":
00127
00128 RANLIB = echo
00129
00130 # But, if possible:
00131
00132 RANLIB = ranlib
00133
00134 # 8. Valgrind's memcheck options (optional):
00135 #
00136
00137 MEMCHECK_OPTS = -v --tool=memcheck --leak-check=full --show-leak-kinds=all \
00138 --track-origins=yes --freelist-vol=20000000
00139
00140 # Done! User, please, do not mess with the definitions from this point on.
00141
00142 #
00143 #
00144 #
00145
00146 # MTK-related.
00147 #
00148
00149 SRC = $(BASE)/src
00150 INCLUDE = $(BASE)/include
00151 LIB = $(BASE)/lib
00152 MTK_LIB = $(LIB)/libmtk.a
00153 TESTS = $(BASE)/tests
00154 EXAMPLES = $(BASE)/examples
00155
00156 # Compiling-related.
00157 #
00158
00159 CCFLAGS += -std=c++11 -fPIC \
00160 -DMTK_VERBOSE_LEVEL=$(VERBOSE_LEVEL) -I$(INCLUDE) -c
00161
00162 ifeq ($(PRECISION),DOUBLE)
00163 CCFLAGS += -DMTK_PRECISION_DOUBLE
00164 else
00165 CCFLAGS += -DMTK_PRECISION_SINGLE
00166 endif
00167
00168 ifeq ($(PERFORM_PREVENTIONS),YES)
00169 CCFLAGS += -DMTK_PERFORM_PREVENTIONS
00170 endif
00171
00172 ifeq ($(VERBOSE_WEIGHTS),YES)
00173 CCFLAGS += -DMTK_VERBOSE_WEIGHTS
00174 endif
00175
00176 # Only the GLPK is included because the other dependencies are coded in Fortran.
00177
00178 ifeq ($(ATL_OPT),ON)
00179 CCFLAGS += -I$(GLPK_INC) $(ATLAS_INC)
00180 else
00181 CCFLAGS += -I$(GLPK_INC)
00182 endif
00183
00184 # Linking-related.
00185 #
00186
00187 NOOPT_LIBS = $(LAPACK_LIB) $(BLAS_LIB) -lm $(GLPK_LIB) -lstdc++
00188
00189 OPT_LIBS = -L$(ATLAS_LIB) -latlas -llapack -lblas -lm -latlas -lstdc++
00190

```

```

00191 ifeq ($(PLAT),OSX)
00192     LINKER = g++
00193     LINKER += -framework Accelerate $(GLPK_LIB) $(MTK_LIB)
00194 else
00195     ifeq ($(ATL_OPT),ON)
00196         LINKER = g++
00197         LIBS = $(MTK_LIB)
00198         LIBS += $(OPT_LIBS)
00199     else
00200         LINKER = gfortran
00201         LIBS = $(MTK_LIB)
00202         LIBS += $(NOOPT_LIBS)
00203     endif
00204 endif
00205
00206 # Documentation-related.
00207 # -----
00208
00209 DOCGEN      = doxygen
00210 DOCFILENAME = doc_config.dxcf
00211 DOC         = $(BASE)/doc
00212 DOCFILE     = $(BASE)/$(DOCFILENAME)

```

18.77 README.md File Reference

18.78 README.md

```

00001 # The Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu**
00004
00005 ## 1. Description
00006
00007 We define numerical methods that are based on discretizations preserving the
00008 properties of their continuous counterparts to be mimetic.
00009
00010 The Mimetic Methods Toolkit (MTK) is a C++11 library for mimetic numerical
00011 methods. It is a set of classes for mimetic interpolation, mimetic
00012 quadratures, and mimetic finite difference methods for the numerical
00013 solution of ordinary and partial differential equations.
00014
00015 ## 2. Dependencies
00016
00017 This README file assumes all of these dependencies are installed in the
00018 following folder:
00019
00020 ```
00021 $(HOME)/Libraries/
00022 ```
00023
00024 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00025 routines for the internal computation on some of the layers. However, ATLAS
00026 requires both BLAS and LAPACK in order to create their optimized distributions.
00027 Therefore, the following dependencies tree arises:
00028
00029 ### For Linux:
00030
00031 1. LAPACK - Available from: http://www.netlib.org/lapack/
00032 1. BLAS - Available from: http://www.netlib.org/blas/
00033
00034 2. GLPK - Available from: https://www.gnu.org/software/glpk/
00035
00036 3. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00037 1. LAPACK - Available from: http://www.netlib.org/lapack/
00038 1. BLAS - Available from: http://www.netlib.org/blas
00039
00040 4. (Optional) Valgrind - Available from: http://valgrind.org/
00041
00042 5. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00043
00044 ### For OS X:
00045
00046 1. GLPK - Available from: https://www.gnu.org/software/glpk/
00047
00048 ## 3. Installation

```



```

00049
00050 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00051
00052 The following steps are required to build and test the MTK. Please use the
00053 accompanying 'Makefile.inc' file, which should provide a solid template to
00054 start with. The following command provides help on the options for make:
00055
00056 ```
00057 $ make help
00058 -----
00059 Makefile for the MTK.
00060
00061 Options are:
00062 - all: builds the library, the tests, and examples.
00063 - mtklib: builds the library.
00064 - test: builds the test files.
00065 - example: builds the examples.
00066
00067 - testall: runs all the tests.
00068
00069 - gendoc: generates the documentation for the library.
00070
00071 - clean: cleans all the generated files.
00072 - cleanlib: cleans the generated archive and object files.
00073 - cleantest: cleans the generated tests executables.
00074 - cleanexample: cleans the generated examples executables.
00075 -----
00076 ```
00077
00078 ### PART 2. BUILD THE LIBRARY.
00079
00080 ```
00081 $ make
00082 ```
00083
00084 If successful you'll read (before building the tests and examples):
00085 ```
00086 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00087 ```
00088
00089 ## 4. Contact, Support, and Credits
00090
00091 The GitHub repository is: https://github.com/ejspeiro/MTK
00092
00093 The MTK is developed by researchers and adjuncts to the
00094 [Computational Science Research Center (CSRC)] (http://www.csrc.sdsu.edu/)
00095 at [San Diego State University (SDSU)] (http://www.sdsu.edu/).
00096
00097 Currently the developers are:
00098
00099 - **Eduardo J. Sanchez, PhD - esanchez@mail.sdsu.edu - @ejspeiro
00100 - Jose E. Castillo, PhD - jcastillo@mail.sdsu.edu
00101 - Guillermo F. Miranda, PhD - unigrav@hotmail.com
00102 - Christopher P. Paolini, PhD - paolini@engineering.sdsu.edu
00103 - Angel Boada.
00104 - Johnny Corbino.
00105 - Raul Vargas-Navarro.
00106
00107 ### 4.1. Acknowledgements and Contributions
00108
00109 The authors would like to acknowledge valuable advising, feedback,
00110 and actual contributions from research personnel at the Computational Science
00111 Research Center (CSRC) at San Diego State University (SDSU). Their input was
00112 important to the fruition of this work. Specifically, our thanks go to
00113 (alphabetical order):
00114
00115 - Mohammad Abouali, PhD
00116 - Dany De Cecchis, PhD
00117 - Otilio Rojas, PhD
00118 - Julia Rossi.
00119
00120 ## 5. Referencing This Work
00121
00122 Please reference this work as follows:
00123 ```
00124 @article{Sanchez2014308,
00125   title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
00126   Finite Differences ",
00127   journal = "Journal of Computational and Applied Mathematics ",
00128   volume = "270",
00129   number = "",

```

```

00130     pages = "308 - 322",
00131     year = "2014",
00132     note = "Fourth International Conference on Finite Element Methods in
00133 Engineering and Sciences (FEMTEC 2013) ",
00134     issn = "0377-0427",
00135     doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
00136     url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
00137     author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
00138     keywords = "Object-oriented development",
00139     keywords = "Partial differential equations",
00140     keywords = "Application programming interfaces",
00141     keywords = "Mimetic Finite Differences "
00142 }
00143
00144 @Inbook{Sanchez2015,
00145     author="Sanchez, Eduardo and Paolini, Christopher and Blomgren, Peter
00146 and Castillo, Jose",
00147     editor="Kirby, M. Robert and Berzins, Martin and Hesthaven, S. Jan",
00148     chapter="Algorithms for Higher-Order Mimetic Operators",
00149     title="Spectral and High Order Methods for Partial Differential Equations
00150 ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014,
00151 Salt Lake City, Utah, USA",
00152     year="2015",
00153     publisher="Springer International Publishing",
00154     address="Cham",
00155     pages="425--434",
00156     isbn="978-3-319-19800-2",
00157     doi="10.1007/978-3-319-19800-2_39",
00158     url="http://dx.doi.org/10.1007/978-3-319-19800-2_39"
00159 }
00160 ```
00161
00162 Finally, please feel free to contact me with suggestions or corrections:
00163
00164 **Eduardo J. Sanchez, PhD - esanchez at mail dot sdsu dot edu** - @ejspeiro
00165
00166 Thanks and happy coding!

```

18.79 src/mtk_blas_adapter.cc File Reference

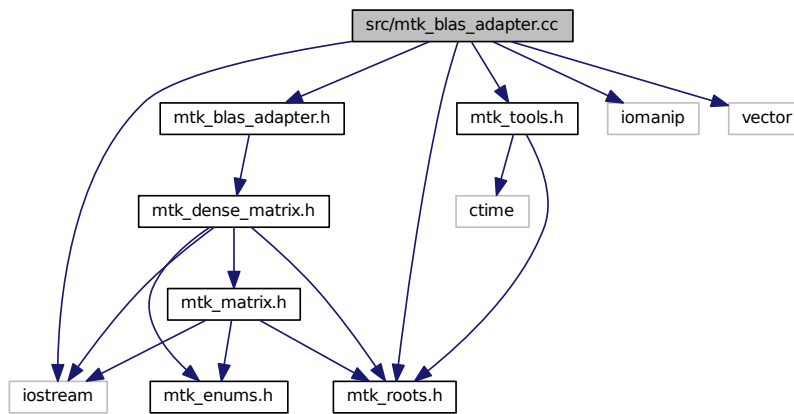
Adapter class for the BLAS API.

```

#include <iostream>
#include <iomanip>
#include <vector>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"

```

Include dependency graph for mtk_blas_adapter.cc:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

- float [mtk::snrm2_](#) (int *n, float *x, int *incx)
- void [mtk::saxpy_](#) (int *n, float *sa, float *sx, int *incx, float *sy, int *incy)
- void [mtk::sgemv_](#) (char *trans, int *m, int *n, float *alpha, float *a, int *lda, float *x, int *incx, float *beta, float *y, int *incy)
- void [mtk::sgemm_](#) (char *transa, char *transb, int *m, int *n, int *k, double *alpha, double *a, int *lda, double *b, aamm int *ldb, double *beta, double *c, int *ldc)

18.79.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the BLAS.

The **BLAS (Basic Linear Algebra Subprograms)** are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.

The BLAS can be installed from links given in the See Also section of this page.

See also

<http://www.netlib.org/blas/>
<https://software.intel.com/en-us/non-commercial-software-development>

Todo Write documentation using LaTeX.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_blas_adapter.cc](#).

18.80 mtk_blas_adapter.cc

```

00001
00027 /*
00028 Copyright (C) 2015, Computational Science Research Center, San Diego State
00029 University. All rights reserved.
00030
00031 Redistribution and use in source and binary forms, with or without modification,
00032 are permitted provided that the following conditions are met:
00033
00034 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00035 and a copy of the modified files should be reported once modifications are
00036 completed, unless these modifications are made through the project's GitHub
00037 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00038 should be developed and included in any deliverable.
00039
00040 2. Redistributions of source code must be done through direct
00041 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00042
00043 3. Redistributions in binary form must reproduce the above copyright notice,
00044 this list of conditions and the following disclaimer in the documentation and/or
00045 other materials provided with the distribution.
00046
00047 4. Usage of the binary form on proprietary applications shall require explicit
00048 prior written permission from the the copyright holders, and due credit should
00049 be given to the copyright holders.
00050
00051 5. Neither the name of the copyright holder nor the names of its contributors
00052 may be used to endorse or promote products derived from this software without
00053 specific prior written permission.
00054
00055 The copyright holders provide no reassurances that the source code provided does
00056 not infringe any patent, copyright, or any other intellectual property rights of
00057 third parties. The copyright holders disclaim any liability to any recipient for
00058 claims brought against recipient by any third party for infringement of that
00059 parties intellectual property rights.
00060
00061 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00062 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00063 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00064 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00065 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00066 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00067 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00068 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00069 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00070 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00071 */
00072
00073 #include <iostream>
00074 #include <iomanip>
00075
00076 #include <vector>
00077
00078 #include "mtk_roots.h"
00079 #include "mtk_tools.h"
00080 #include "mtk_blas_adapter.h"
00081
00082 namespace mtk {
00083
00084 extern "C" {
00085
00086 #ifdef MTK_PRECISION_DOUBLE
00087
00100 double dnm2_(int *n, double *x, int *incx);
00101 #else
00102
00115 float snrm2_(int *n, float *x, int *incx);
00116 #endif
00117

```

```

00118 #ifdef MTK_PRECISION_DOUBLE
00119
00138 void daxpy_(int *n, double *da, double *dx, int *incx, double *dy, int *incy);
00139 #else
00140
00159 void saxpy_(int *n, float *sa, float *sx, int *incx, float *sy, int *incy);
00160 #endif
00161
00162 #ifdef MTK_PRECISION_DOUBLE
00163
00191 void dgemv_(char *trans,
00192             int *m,
00193             int *n,
00194             double *alpha,
00195             double *a,
00196             int *lda,
00197             double *x,
00198             int *incx,
00199             double *beta,
00200             double *y,
00201             int *incy);
00202 #else
00203
00231 void sgemv_(char *trans,
00232             int *m,
00233             int *n,
00234             float *alpha,
00235             float *a,
00236             int *lda,
00237             float *x,
00238             int *incx,
00239             float *beta,
00240             float *y,
00241             int *incy);
00242 #endif
00243
00244 #ifdef MTK_PRECISION_DOUBLE
00245
00270 void dgemm_(char *transa,
00271             char* transb,
00272             int *m,
00273             int *n,
00274             int *k,
00275             double *alpha,
00276             double *a,
00277             int *lda,
00278             double *b,
00279             int *ldb,
00280             double *beta,
00281             double *c,
00282             int *ldc);
00283 }
00284 #else
00285
00310 void sgemm_(char *transa,
00311             char* transb,
00312             int *m,
00313             int *n,
00314             int *k,
00315             double *alpha,
00316             double *a,
00317             int *lda,
00318             double *b, aamm
00319             int *ldb,
00320             double *beta,
00321             double *c,
00322             int *ldc);
00323 }
00324 #endif
00325 }
00326
00327 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00328
00329 #ifdef MTK_PERFORM_PREVENTIONS
00330 mtk::Tools::Prevent(in_length <= 0, __FILE__, __LINE__, __func__);
00331 #endif
00332
00333 int incx{1}; // Increment for the elements of xx. ix >= 0.
00334
00335 #ifdef MTK_PRECISION_DOUBLE
00336 return dnrn2_(&in_length, in, &incx);

```

```

00337     #else
00338     return snrm2_(&in_length, in, &incx);
00339     #endif
00340 }
00341
00342 void mtk::BLASAdapter::RealAXPY(mtk::Real alpha,
00343                                 mtk::Real *xx,
00344                                 mtk::Real *yy,
00345                                 int &in_length) {
00346
00347     #ifdef MTK_PERFORM_PREVENTIONS
00348     mtk::Tools::Prevent(xx == nullptr, __FILE__, __LINE__, __func__);
00349     mtk::Tools::Prevent(yy == nullptr, __FILE__, __LINE__, __func__);
00350     #endif
00351
00352     int incx{1}; // Increment for the elements of xx. ix >= 0.
00353
00354     #ifdef MTK_PRECISION_DOUBLE
00355     daxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00356     #else
00357     saxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00358     #endif
00359 }
00360
00361 mtk::Real mtk::BLASAdapter::RelNorm2Error(
00362     mtk::Real *computed,
00363     mtk::Real *known,
00364     int length) {
00365
00366     #ifdef MTK_PERFORM_PREVENTIONS
00367     mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00368     mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00369     #endif
00370
00371     mtk::Real norm_2_computed{mtk::BLASAdapter::RealNRM2(known, length)};
00372
00373     mtk::Real alpha{-mtk::kOne};
00374
00375     mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00376
00377     mtk::Real norm_2_difference{mtk::BLASAdapter::RealNRM2(computed,
00378     length)};
00379
00380     return norm_2_difference/norm_2_computed;
00381 }
00382
00383 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00384                                     mtk::DenseMatrix &aa,
00385                                     mtk::Real *xx,
00386                                     mtk::Real &beta,
00387                                     mtk::Real *yy) {
00388
00389     // Make sure input matrices are row-major ordered.
00390
00391     if (aa.matrix_properties().ordering() ==
00392         mtk::MatrixOrdering::COL_MAJOR) {
00393         aa.OrderRowMajor();
00394     }
00395
00396     char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00397
00398     int mm{aa.num_rows()}; // Rows of aa.
00399     int nn{aa.num_cols()}; // Columns of aa.
00400     int lda{(aa.matrix_properties()).ld()}; // Leading dimension.
00401     int incx{1}; // Increment of values in x.
00402     int incy{1}; // Increment of values in y.
00403
00404     std::swap(mm, nn);
00405     #ifdef MTK_PRECISION_DOUBLE
00406     dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00407           xx, &incx, &beta, yy, &incy);
00408     #else
00409     sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00410           xx, &incx, &beta, yy, &incy);
00411     #endif
00412     std::swap(mm, nn);
00413 }
00414
00415 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
00416     mtk::DenseMatrix &aa,
00417     mtk::DenseMatrix &bb) {

```

```

00414
00415 #ifdef MTK_PERFORM_PREVENTIONS
00416 mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00417                     __FILE__, __LINE__, __func__);
00418 #endif
00419
00421 if (aa.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00422     aa.OrderRowMajor();
00423 }
00424 if (bb.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00425     bb.OrderRowMajor();
00426 }
00427
00429 char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00430 char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00431
00432 int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00433 int nn{bb.num_cols()}; // Cols of bb and cols of cc.
00434 int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00435
00436 int cc_num_rows{mm}; // Rows of cc.
00437 int cc_num_cols{nn}; // Columns of cc.
00438
00439 int lda{std::max(1, kk)}; // Leading dimension of the aa matrix.
00440 int ldb{std::max(1, nn)}; // Leading dimension of the bb matrix.
00441 int ldc{std::max(1, mm)}; // Leading dimension of the cc matrix.
00442
00443 mtk::Real alpha{mtk::kOne}; // First scalar coefficient.
00444 mtk::Real beta{mtk::kZero}; // Second scalar coefficient.
00445
00446 mtk::DenseMatrix cc_col_maj_ord(cc_num_rows, cc_num_cols); // Output matrix.
00447
00448 cc_col_maj_ord.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00449
00451 #ifdef MTK_PRECISION_DOUBLE
00452 dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00453        bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00454 #else
00455 sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00456        bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00457 #endif
00458
00459 #if MTK_VERBOSE_LEVEL > 12
00460 std::cout << "cc_col_maj_ord =" << std::endl;
00461 std::cout << cc_col_maj_ord << std::endl;
00462 #endif
00463
00464 cc_col_maj_ord.OrderRowMajor();
00465
00466 return cc_col_maj_ord;
00467 }
00468
00469 mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM(
mtk::Real alpha,
00470
00471                                     mtk::DenseMatrix &aa) {
00472
00473 #ifdef MTK_PERFORM_PREVENTIONS
00474 mtk::Tools::Prevent(aa.num_rows() == 0, __FILE__, __LINE__, __func__);
00475 mtk::Tools::Prevent(aa.num_cols() == 0, __FILE__, __LINE__, __func__);
00476 #endif
00477
00478 if (aa.matrix_properties().ordering() ==
mtk::MatrixOrdering::COL_MAJOR) {
00479     aa.OrderRowMajor();
00480 }
00481
00483 char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00484 char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00485
00486 int mm{aa.num_rows()}; // Rows of aa and rows of cc.
00487 int nn{aa.num_cols()}; // Cols of bb and cols of cc.
00488 int kk{aa.num_cols()}; // Cols of aa and rows of bb.
00489
00490 int lda{std::max(1, kk)}; // Leading dimension of the aa matrix.
00491 int ldb{std::max(1, nn)}; // Leading dimension of the bb matrix.
00492 int ldc{std::max(1, mm)}; // Leading dimension of the cc matrix.
00493
00494 mtk::Real beta{alpha}; // Second scalar coefficient.
00495

```

```

00496   alpha = mtk::kZero;
00497
00498   mtk::DenseMatrix alpha_aa(aa); // Output matrix.
00499
00501   #ifdef MTK_PRECISION_DOUBLE
00502   dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00503         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00504   #else
00505   sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00506         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00507   #endif
00508
00509   #if MTK_VERBOSE_LEVEL > 12
00510   std::cout << "alpha_aa =" << std::endl;
00511   std::cout << alpha_aa << std::endl;
00512   #endif
00513
00514   return alpha_aa;
00515 }

```

18.81 src/mtk_curl_2d.cc File Reference

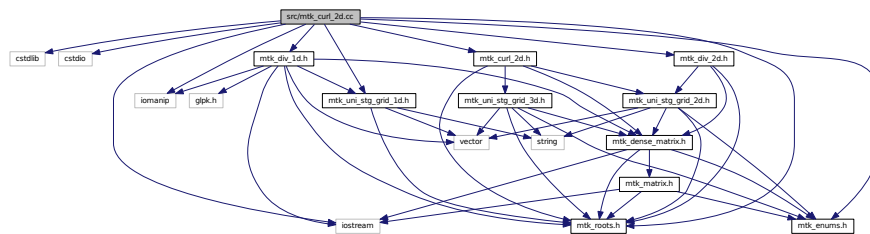
Implements the class Curl2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"
#include "mtk_curl_2d.h"

```

Include dependency graph for mtk_curl_2d.cc:



18.81.1 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](#).

18.82 mtk_curl_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_enums.h"
00065 #include "mtk_uni_stg_grid_1d.h"
00066 #include "mtk_div_1d.h"
00067 #include "mtk_div_2d.h"
00068 #include "mtk_curl_2d.h"
00069
00070 mtk::UniStgGrid3D mtk::Curl2D::operator*(const
    mtk::UniStgGrid2D &grid) const {
00071
00072
00073     mtk::UniStgGrid3D output;
00074
00075     return output;
00076 }
00077
00078
00079 mtk::Curl2D::Curl2D():
00080     order_accuracy_(0),
00081     mimetic_threshold_(0) {}
00082
00083 mtk::Curl2D::Curl2D(const Curl2D &curl):
00084     order_accuracy_(curl.order_accuracy_),
00085     mimetic_threshold_(curl.mimetic_threshold_) {}
00086
00087 mtk::Curl2D::~Curl2D() {}

```

```

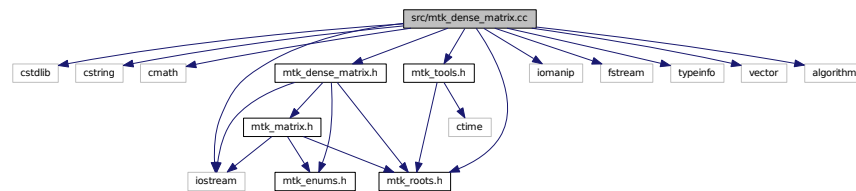
00088
00089 bool mtk::Curl2D::ConstructCurl2D(const
    mtk::UniStgGrid2D &grid,
00090                                     int order_accuracy,
00091                                     mtk::Real mimetic_threshold) {
00092
00093     int num_cells_x = grid.num_cells_x();
00094     int num_cells_y = grid.num_cells_y();
00095
00096     int mx = num_cells_x + 2; // Dx vertical dimension.
00097     int nx = num_cells_x + 1; // Dx horizontal dimension.
00098     int my = num_cells_y + 2; // Dy vertical dimension.
00099     int ny = num_cells_y + 1; // Dy horizontal dimension.
00100
00101     mtk::Div1D div;
00102
00103     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00104
00105     #ifdef MTK_PERFORM_PREVENTIONS
00106     if (!info) {
00107         std::cerr << "Mimetic div could not be built." << std::endl;
00108         return info;
00109     }
00110     #endif
00111
00112     auto west = grid.west_bndy();
00113     auto east = grid.east_bndy();
00114     auto south = grid.south_bndy();
00115     auto north = grid.east_bndy();
00116
00117     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00118     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00119
00120     mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00121     mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00122
00123     bool padded{true};
00124     bool transpose{false};
00125
00126     mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00127     mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00128
00129     mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00130     mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00131
00132     #if MTK_VERBOSE_LEVEL > 2
00133     std::cout << "Dx: " << mx << " by " << nx << std::endl;
00134     std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;
00135     std::cout << "Dy: " << my << " by " << ny << std::endl;
00136     std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00137     std::cout << "Curl 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
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     return info;
00155 }
00156
00157 mtk::DenseMatrix mtk::Curl2D::ReturnAsDenseMatrix() const {
00158
00159     return curl_;
00160 }

```

18.83 src/mtk_dense_matrix.cc File Reference

```
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <typeinfo>
#include <vector>
#include <algorithm>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_tools.h"
```

Include dependency graph for mtk_dense_matrix.cc:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::DenseMatrix &in)`

18.84 mtk_dense_matrix.cc

```
00001
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
```

```

00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #include <cstdlib>
00060 #include <cstring>
00061 #include <cmath>
00062
00063 #include <iostream>
00064 #include <iomanip>
00065 #include <fstream>
00066
00067 #include <typeinfo>
00068
00069 #include <vector>
00070
00071 #include <algorithm>
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_dense_matrix.h"
00075 #include "mtk_tools.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00080
00081     int mm{in.matrix_properties_.num_rows()}; // Auxiliary.
00082     int nn{in.matrix_properties_.num_cols()}; // Auxiliary.
00083     int output_precision{4};
00084     int output_width{10};
00085
00086     if (in.matrix_properties_.ordering() ==
00087         mtk::MatrixOrdering::COL_MAJOR) {
00088         std::swap(mm, nn);
00089     }
00090     for (int ii = 0; ii < mm; ii++) {
00091         int offset(ii*nn);
00092         for (int jj = 0; jj < nn; jj++) {
00093             mtk::Real value = in.data_[offset + jj];
00094             stream << std::setprecision(output_precision) <<
00095                 std::setw(output_width) << value;
00096         }
00097         stream << std::endl;
00098     }
00099     if (in.matrix_properties_.ordering() ==
00100         mtk::MatrixOrdering::COL_MAJOR) {
00101         std::swap(mm, nn);
00102     }
00103     return stream;
00104 }
00105 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
00106 mtk::DenseMatrix &in) {
00107
00108     if(this == &in) {
00109         return *this;
00110     }

```

```

00110
00111     matrix_properties_.set_storage(in.
matrix_properties_.storage());
00112
00113     matrix_properties_.set_ordering(in.
matrix_properties_.ordering());
00114
00115     auto aux = in.matrix_properties_.num_rows();
00116     matrix_properties_.set_num_rows(aux);
00117
00118     aux = in.matrix_properties().num_cols();
00119     matrix_properties_.set_num_cols(aux);
00120
00121     aux = in.matrix_properties().num_zero();
00122     matrix_properties_.set_num_zero(aux);
00123
00124     aux = in.matrix_properties().num_null();
00125     matrix_properties_.set_num_null(aux);
00126
00127     auto num_rows = matrix_properties_.num_rows();
00128     auto num_cols = matrix_properties_.num_cols();
00129
00130     delete [] data_;
00131
00132     try {
00133         data_ = new mtk::Real[num_rows*num_cols];
00134     } catch (std::bad_alloc &memory_allocation_exception) {
00135         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00136             std::endl;
00137         std::cerr << memory_allocation_exception.what() << std::endl;
00138     }
00139     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
num_cols);
00140
00141     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00142
00143     return *this;
00144 }
00145
00146 bool mtk::DenseMatrix::operator ==(const
DenseMatrix &in) {
00147
00148     bool ans{true};
00149
00150     auto mm = in.num_rows();
00151     auto nn = in.num_cols();
00152
00153     if (mm != matrix_properties_.num_rows() ||
00154         nn != matrix_properties_.num_cols()) {
00155         return false;
00156     }
00157
00158     for (int ii = 0; ii < mm && ans; ++ii) {
00159         for (int jj = 0; jj < nn && ans; ++jj) {
00160             ans = ans &&
00161                 abs(data_[ii*nn + jj] - in.data()[ii*nn + jj]) <
mtk::kDefaultTolerance;
00162         }
00163     }
00164     return ans;
00165 }
00166
00167 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00168
00169     matrix_properties_.set_storage(
mtk::MatrixStorage::DENSE);
00170     matrix_properties_.set_ordering(
mtk::MatrixOrdering::ROW_MAJOR);
00171 }
00172
00173 mtk::DenseMatrix::DenseMatrix(const
mtk::DenseMatrix &in) {
00174
00175     matrix_properties_.set_storage(in.matrix_properties_.storage());
00176
00177     matrix_properties_.set_ordering(in.matrix_properties_.
ordering());
00178
00179     auto aux = in.matrix_properties_.num_rows();
00180     matrix_properties_.set_num_rows(aux);
00181

```

```

00182     aux = in.matrix_properties().num_cols();
00183     matrix_properties_.set_num_cols(aux);
00184
00185     aux = in.matrix_properties().num_zero();
00186     matrix_properties_.set_num_zero(aux);
00187
00188     aux = in.matrix_properties().num_null();
00189     matrix_properties_.set_num_null(aux);
00190
00191     auto num_rows = in.matrix_properties_.num_rows();
00192     auto num_cols = in.matrix_properties_.num_cols();
00193
00194     try {
00195         data_ = new mtk::Real[num_rows*num_cols];
00196     } catch (std::bad_alloc &memory_allocation_exception) {
00197         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00198             std::endl;
00199         std::cerr << memory_allocation_exception.what() << std::endl;
00200     }
00201     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00202
00203     std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00204 }
00205
00206 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00207
00208     #ifdef MTK_PERFORM_PREVENTIONS
00209     mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00210     mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00211     #endif
00212
00213     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00214     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00215     matrix_properties_.set_num_rows(num_rows);
00216     matrix_properties_.set_num_cols(num_cols);
00217
00218     try {
00219         data_ = new mtk::Real[num_rows*num_cols];
00220     } catch (std::bad_alloc &memory_allocation_exception) {
00221         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00222             std::endl;
00223         std::cerr << memory_allocation_exception.what() << std::endl;
00224     }
00225     memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00226 }
00227
00228 mtk::DenseMatrix::DenseMatrix(const int &rank,
00229                               const bool &padded,
00230                               const bool &transpose) {
00231
00232     #ifdef MTK_PERFORM_PREVENTIONS
00233     mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00234     #endif
00235
00236     int aux{}; // Used to control the padding.
00237
00238     if (padded) {
00239         aux = 1;
00240     }
00241
00242     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00243     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00244     matrix_properties_.set_num_rows(aux + rank + aux);
00245     matrix_properties_.set_num_cols(rank);
00246
00247     try {
00248         data_ = new mtk::Real[matrix_properties_.num_values()];
00249     } catch (std::bad_alloc &memory_allocation_exception) {
00250         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00251             std::endl;
00252         std::cerr << memory_allocation_exception.what() << std::endl;
00253     }
00254     memset(data_,
00255            mtk::kZero,
00256            sizeof(data_[0])*(matrix_properties_.num_values()));
00257
00258     for (auto ii = 0; ii < matrix_properties_.num_rows(); ++ii) {
00259         for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00260             data_[ii*matrix_properties_.num_cols() + jj] =
00261                 (ii == jj + aux)? mtk::kOne : mtk::kZero;
00262         }
00263     }

```

```

00263     }
00264     if (transpose) {
00265         Transpose();
00266     }
00267 }
00268
00269 mtk::DenseMatrix::DenseMatrix(const mtk::Real *const gen,
00270                               const int &gen_length,
00271                               const int &pro_length,
00272                               const bool &transpose) {
00273
00274     #ifdef MTK_PERFORM_PREVENTIONS
00275     mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00276     mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00277     mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00278     #endif
00279
00280     matrix_properties_.set_storage(mtk::MatrixStorage::DENSE);
00281     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00282     if (!transpose) {
00283         matrix_properties_.set_num_rows(gen_length);
00284         matrix_properties_.set_num_cols(pro_length);
00285     } else {
00286         matrix_properties_.set_num_rows(pro_length);
00287         matrix_properties_.set_num_cols(gen_length);
00288     }
00289
00290     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00291     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00292
00293     try {
00294         data_ = new mtk::Real[mm*nn];
00295     } catch (std::bad_alloc &memory_allocation_exception) {
00296         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00297             std::endl;
00298         std::cerr << memory_allocation_exception.what() << std::endl;
00299     }
00300     memset(data_, mtk::kZero, sizeof(data_[0])*mm*nn);
00301
00302     if (!transpose) {
00303         for (auto ii = 0; ii < mm; ii++) {
00304             for (auto jj = 0; jj < nn; jj++) {
00305                 data_[ii*nn + jj] = pow(gen[ii], (double) jj);
00306             }
00307         }
00308     } else {
00309         for (auto ii = 0; ii < mm; ii++) {
00310             for (auto jj = 0; jj < nn; jj++) {
00311                 data_[ii*nn + jj] = pow(gen[jj], (double) ii);
00312             }
00313         }
00314     }
00315 }
00316
00317 mtk::DenseMatrix::~DenseMatrix() {
00318     delete [] data_;
00319     data_ = nullptr;
00320 }
00321
00322
00323 mtk::Matrix mtk::DenseMatrix::matrix_properties() const
00324     noexcept {
00325     return matrix_properties_;
00326 }
00327
00328 void mtk::DenseMatrix::SetOrdering(
00329     mtk::MatrixOrdering oo) noexcept {
00330
00331     #ifdef MTK_PERFORM_PREVENTIONS
00332     mtk::Tools::Prevent(!(oo == mtk::MatrixOrdering::ROW_MAJOR
00333         || oo ==
00334         mtk::MatrixOrdering::COL_MAJOR),
00335         __FILE__, __LINE__, __func__);
00336     #endif
00337     matrix_properties_.set_ordering(oo);
00338 }
00339
00340 int mtk::DenseMatrix::num_rows() const noexcept {

```

```

00341     return matrix_properties_.num_rows();
00342 }
00343
00344 int mtk::DenseMatrix::num_cols() const noexcept {
00345
00346     return matrix_properties_.num_cols();
00347 }
00348
00349 mtk::Real* mtk::DenseMatrix::data() const noexcept {
00350
00351     return data_;
00352 }
00353
00354 mtk::Real mtk::DenseMatrix::GetValue(
00355     const int &mm,
00356     const int &nn) const noexcept {
00357
00358     #ifdef MTK_PERFORM_PREVENTIONS
00359     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00360     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00361     #endif
00362
00363     return data_[mm*matrix_properties_.num_cols() + nn];
00364 }
00365
00366 void mtk::DenseMatrix::SetValue(
00367     const int &mm,
00368     const int &nn,
00369     const mtk::Real &val) noexcept {
00370
00371     #ifdef MTK_PERFORM_PREVENTIONS
00372     mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00373     mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00374     #endif
00375
00376     data_[mm*matrix_properties_.num_cols() + nn] = val;
00377 }
00378
00379 void mtk::DenseMatrix::Transpose() {
00380
00381     mtk::Real *data_transposed{}; // Buffer.
00382
00383     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00384     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00385
00386     try {
00387         data_transposed = new mtk::Real[mm*nn];
00388     } catch (std::bad_alloc &memory_allocation_exception) {
00389         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00390             std::endl;
00391         std::cerr << memory_allocation_exception.what() << std::endl;
00392     }
00393     memset(data_transposed,
00394         mtk::kZero,
00395         sizeof(data_transposed[0])*mm*nn);
00396
00397     // Assign the values to their transposed position.
00398     for (auto ii = 0; ii < mm; ++ii) {
00399         for (auto jj = 0; jj < nn; ++jj) {
00400             data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00401         }
00402     }
00403
00404     // Swap pointers.
00405     auto tmp = data_; // Temporal holder.
00406     data_ = data_transposed;
00407     delete [] tmp;
00408     tmp = nullptr;
00409
00410     matrix_properties_.set_num_rows(nn);
00411     matrix_properties_.set_num_cols(mm);
00412 }
00413
00414 void mtk::DenseMatrix::OrderRowMajor() {
00415
00416     if (matrix_properties_.ordering() == mtk::MatrixOrdering::COL_MAJOR) {
00417
00418         mtk::Real *data_transposed{}; // Buffer.
00419
00420
00421
00422
00423

```



```

00424     int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00425     int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00426
00427     try {
00428         data_transposed = new mtk::Real[mm*nn];
00429     } catch (std::bad_alloc &memory_allocation_exception) {
00430         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00431             std::endl;
00432         std::cerr << memory_allocation_exception.what() << std::endl;
00433     }
00434     memset(data_transposed,
00435         mtk::kZero,
00436         sizeof(data_transposed[0])*mm*nn);
00437
00438     // Assign the values to their transposed position.
00439     std::swap(mm, nn);
00440     for (auto ii = 0; ii < mm; ++ii) {
00441         for (auto jj = 0; jj < nn; ++jj) {
00442             data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00443         }
00444     }
00445     std::swap(mm, nn);
00446
00447     // Swap pointers.
00448     auto tmp = data_; // Temporal holder.
00449     data_ = data_transposed;
00450     delete [] tmp;
00451     tmp = nullptr;
00452
00453     matrix_properties_.set_ordering(mtk::MatrixOrdering::ROW_MAJOR);
00454 }
00455 }
00456
00457 void mtk::DenseMatrix::OrderColMajor() {
00458
00459     if (matrix_properties_.ordering() == mtk::MatrixOrdering::ROW_MAJOR) {
00460
00461         mtk::Real *data_transposed{}; // Buffer.
00462
00463         int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00464         int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00465
00466         try {
00467             data_transposed = new mtk::Real[mm*nn];
00468         } catch (std::bad_alloc &memory_allocation_exception) {
00469             std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00470                 std::endl;
00471             std::cerr << memory_allocation_exception.what() << std::endl;
00472         }
00473         memset(data_transposed,
00474             mtk::kZero,
00475             sizeof(data_transposed[0])*mm*nn);
00476
00477         // Assign the values to their transposed position.
00478         for (auto ii = 0; ii < mm; ++ii) {
00479             for (auto jj = 0; jj < nn; ++jj) {
00480                 data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00481             }
00482         }
00483     }
00484
00485     // Swap pointers.
00486     auto tmp = data_; // Temporal holder.
00487     data_ = data_transposed;
00488     delete [] tmp;
00489     tmp = nullptr;
00490
00491     matrix_properties_.set_ordering(mtk::MatrixOrdering::COL_MAJOR);
00492 }
00493 }
00494 }
00495
00496 mtk::DenseMatrix mtk::DenseMatrix::Kron(const
00497     mtk::DenseMatrix &aa,
00498                                     const mtk::DenseMatrix &bb) {
00499
00500     int row_offset{}; // Offset for rows.
00501     int col_offset{}; // Offset for rows.
00502
00503     mtk::Real aa_factor{}; // Used in computation.
00504
00505

```

```

00506 // Auxiliary variables:
00507 auto aux1 = aa.matrix_properties_.num_rows()*bb.
matrix_properties_.num_rows();
00508 auto aux2 = aa.matrix_properties_.num_cols()*bb.
matrix_properties_.num_cols();
00509
00510 mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00511
00512 int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00513
00514 auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00515 auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00516 auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00517 auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00518
00519 for (auto ii = 0; ii < mm; ++ii) {
00520     row_offset = ii*pp;
00521     for (auto jj = 0; jj < nn; ++jj) {
00522         col_offset = jj*qq;
00523         aa_factor = aa.data_[ii*nn + jj];
00524         for (auto ll = 0; ll < pp; ++ll) {
00525             for (auto oo = 0; oo < qq; ++oo) {
00526                 auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00527                 output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00528             }
00529         }
00530     }
00531 }
00532
00533 output.matrix_properties_.set_storage(
mtk::MatrixStorage::DENSE);
00534 output.matrix_properties_.set_ordering(
mtk::MatrixOrdering::ROW_MAJOR);
00535
00536 return output;
00537 }
00538
00539 bool mtk::DenseMatrix::WriteToFile(const std::string &filename) const {
00540
00541     std::ofstream output_dat_file; // Output file.
00542
00543     output_dat_file.open(filename);
00544
00545     if (!output_dat_file.is_open()) {
00546         return false;
00547     }
00548
00549     int mm{matrix_properties_.num_rows()};
00550     int nn{matrix_properties_.num_cols()};
00551
00552     for (int ii = 0; ii < mm; ++ii) {
00553         int offset{ii*nn};
00554         for (int jj = 0; jj < nn; ++jj) {
00555             output_dat_file << ii << ' ' << jj << ' ' << data_[offset + jj] <<
std::endl;
00556         }
00557     }
00558
00559     output_dat_file.close();
00560
00561     return true;
00562 }
00563 }

```

18.85 src/mtk_div_1d.cc File Reference

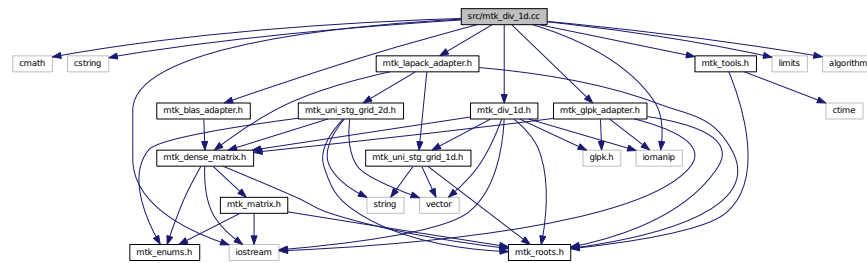
Implements the class Div1D.

```

#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)`

18.85.1 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 `mtk::BLASAdapter`.

Definition in file [mtk_div_1d.cc](#).

18.86 mtk_div_1d.cc

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075
00076 #include "mtk_blas_adapter.h"
00077 #include "mtk_lapack_adapter.h"
00078 #include "mtk_glpk_adapter.h"
00079
00080 #include "mtk_div_1d.h"
00081
00082 namespace mtk {
00083
00084 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00085
00086     int output_precision{4};
00087     int output_width{8};
00088
00089     stream << "Order of accuracy: " << in.divergence_[0] << std::endl;
00090
00091
00092

```

```

00094
00095     stream << "Interior stencil: " << std::endl;
00096     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00097         stream << std::setprecision(output_precision) << std::setw(output_width) <<
00098             in.divergence_[ii] << ' ';
00099     }
00100     stream << std::endl;
00101
00102     if (in.order_accuracy_ > 2) {
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;
00111
00112         auto offset = (2*in.order_accuracy_ + 1);
00113         int mm{};
00114         for (auto ii = 0; ii < in.dim_null_; ++ii) {
00115             stream << "Mimetic boundary row " << ii + 1 << ": " << std::endl;
00116             for (auto jj = 0; jj < in.num_bndy_coefs_; ++jj) {
00117                 auto value = in.divergence_[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 row " << ii + 1 << ": " <<
00124                 in.sums_rows_mim_bndy_[ii];
00125             stream << std::endl;
00126         }
00127     }
00128     return stream;
00129 }
00130
00131
00132 mtk::Div1D::Div1D():
00133     order_accuracy_(mtk::kDefaultOrderAccuracy),
00134     dim_null_(),
00135     num_bndy_coefs_(),
00136     divergence_length_(),
00137     minrow_(),
00138     row_(),
00139     coefs_interior_(),
00140     prem_apps_(),
00141     weights_crs_(),
00142     weights_cbs_(),
00143     mim_bndy_(),
00144     divergence_(),
00145     sums_rows_mim_bndy_(),
00146     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00147
00148 mtk::Div1D::Div1D(const Div1D &div):
00149     order_accuracy_(div.order_accuracy_),
00150     dim_null_(div.dim_null_),
00151     num_bndy_coefs_(div.num_bndy_coefs_),
00152     divergence_length_(div.divergence_length_),
00153     minrow_(div.minrow_),
00154     row_(div.row_),
00155     coefs_interior_(div.coefs_interior_),
00156     prem_apps_(div.prem_apps_),
00157     weights_crs_(div.weights_crs_),
00158     weights_cbs_(div.weights_cbs_),
00159     mim_bndy_(div.mim_bndy_),
00160     divergence_(div.divergence_),
00161     sums_rows_mim_bndy_(div.sums_rows_mim_bndy_),
00162     mimetic_threshold_(div.mimetic_threshold_) {}
00163
00164 mtk::Div1D::~Div1D() {
00165     delete[] coefs_interior_;
00166     coefs_interior_ = nullptr;
00167
00168     delete[] prem_apps_;
00169     prem_apps_ = nullptr;
00170 }

```

```

00176 delete[] weights_crs_;
00177 weights_crs_ = nullptr;
00178
00179 delete[] weights_cbs_;
00180 weights_cbs_ = nullptr;
00181
00182 delete[] mim_bndy_;
00183 mim_bndy_ = nullptr;
00184
00185 delete[] divergence_;
00186 divergence_ = nullptr;
00187 }
00188
00189 bool mtk::Div1D::ConstructDiv1D(int order_accuracy,
00190                                mtk::Real mimetic_threshold) {
00191
00192     #ifdef MTK_PERFORM_PREVENTIONS
00193     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00194     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00195     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00196                         __FILE__, __LINE__, __func__);
00197
00198     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00199         std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00200     }
00201
00202     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00203     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00204     #endif
00205
00206     order_accuracy_ = order_accuracy;
00207     mimetic_threshold_ = mimetic_threshold;
00208
00209
00210
00211     bool abort_construction = ComputeStencilInteriorGrid();
00212
00213     #ifdef MTK_PERFORM_PREVENTIONS
00214     if (!abort_construction) {
00215         std::cerr << "Could NOT complete stage 1." << std::endl;
00216         std::cerr << "Exiting..." << std::endl;
00217         return false;
00218     }
00219     #endif
00220
00221     // At this point, we already have the values for the interior stencil stored
00222     // in the coeffs_interior_ array.
00223
00224     // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00225     // approximation at the boundary, thus it has no weights. For this case, the
00226     // dimension of the null-space of the Vandermonde matrices used to compute the
00227     // approximating coefficients at the boundary is 0. Ergo, we compute this
00228     // number first and then decide if we must compute anything at the boundary.
00229
00230     dim_null_ = order_accuracy_/2 - 1;
00231
00232     if (dim_null_ > 0) {
00233
00234         #ifdef MTK_PRECISION_DOUBLE
00235         num_bndy_coeffs_ = (int) (3.0*(mtk::Real) order_accuracy_)/2.0);
00236         #else
00237         num_bndy_coeffs_ = (int) (3.0f*(mtk::Real) order_accuracy_)/2.0f);
00238         #endif
00239
00240
00241         // For this we will follow recommendations given in:
00242         //
00243         // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00244         //
00245         // We will compute the QR Factorization of the transpose, as in the
00246         // following (MATLAB) pseudo-code:
00247         //
00248         // [Q,R] = qr(V'); % Full QR as defined in
00249         // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00250         //
00251         // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00252         //
00253         // However, given the nature of the Vandermonde matrices we've just
00254         // computed, they all possess the same null-space. Therefore, we impose the
00255         // convention of computing the null-space of the first Vandermonde matrix
00256         // (west boundary).
00257
00258

```

```

00259     abort_construction = ComputeRationalBasisNullSpace();
00260
00261     #ifdef MTK_PERFORM_PREVENTIONS
00262     if (!abort_construction) {
00263         std::cerr << "Could NOT complete stage 2.1." << std::endl;
00264         std::cerr << "Exiting..." << std::endl;
00265         return false;
00266     }
00267     #endif
00268
00270
00271     abort_construction = ComputePreliminaryApproximations();
00272
00273     #ifdef MTK_PERFORM_PREVENTIONS
00274     if (!abort_construction) {
00275         std::cerr << "Could NOT complete stage 2.2." << std::endl;
00276         std::cerr << "Exiting..." << std::endl;
00277         return false;
00278     }
00279     #endif
00280
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
00294
00295     abort_construction = ComputeStencilBoundaryGrid();
00296
00297     #ifdef MTK_PERFORM_PREVENTIONS
00298     if (!abort_construction) {
00299         std::cerr << "Could NOT complete stage 2.4." << std::endl;
00300         std::cerr << "Exiting..." << std::endl;
00301         return false;
00302     }
00303     #endif
00304
00305 } // End of: if (dim_null_ > 0);
00306
00308
00309 // Once we have the following three collections of data:
00310 //   (a) the coefficients for the interior,
00311 //   (b) the coefficients for the boundary (if it applies),
00312 //   (c) and the weights (if it applies),
00313 // we will store everything in the output array:
00314
00315 abort_construction = AssembleOperator();
00316
00317 #ifdef MTK_PERFORM_PREVENTIONS
00318 if (!abort_construction) {
00319     std::cerr << "Could NOT complete stage 3." << std::endl;
00320     std::cerr << "Exiting..." << std::endl;
00321     return false;
00322 }
00323 #endif
00324
00325 return true;
00326 }
00327
00328 int mtk::Div1D::num_bndy_coeffs() const {
00329
00330     return num_bndy_coeffs_;
00331 }
00332
00333 mtk::Real *mtk::Div1D::coeffs_interior() const {
00334
00335     return coeffs_interior_;
00336 }
00337
00338 mtk::Real *mtk::Div1D::weights_crs() const {
00339
00340     return weights_crs_;
00341 }
00342
00343 mtk::Real *mtk::Div1D::weights_cbs() const {

```

```

00344
00345     return weights_cbs_;
00346 }
00347
00348 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00349
00350     mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00351
00352     auto counter = 0;
00353     for (auto ii = 0; ii < dim_null_; ++ii) {
00354         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00355             xx.SetValue(ii,jj, divergence_[2*order_accuracy_ + 1 + counter]);
00356             counter++;
00357         }
00358     }
00359     return xx;
00360 }
00361 }
00362
00363 std::vector<mtk::Real> mtk::Div1D::sums_rows_mim_bndy() const {
00364
00365     return sums_rows_mim_bndy_;
00366 }
00367
00368 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(
00369     const UniStgGrid1D &grid) const {
00370
00371     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00372
00373     #ifdef MTK_PERFORM_PREVENTIONS
00374     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00375     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00376     #endif
00377
00378     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00379
00380     int dd_num_rows = nn + 2;
00381     int dd_num_cols = nn + 1;
00382     int elements_per_row = num_bndy_coeffs_;
00383     int num_extra_rows = dim_null_;
00384
00385     // Output matrix featuring sizes for divergence operators.
00386     mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00387
00388
00389     auto ee_index = 0;
00390     for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00391         auto cc = 0;
00392         for (auto jj = 0; jj < dd_num_rows; jj++) {
00393             if (cc >= elements_per_row) {
00394                 out.SetValue(ii, jj, mtk::kZero);
00395             } else {
00396                 out.SetValue(ii, jj, mim_bndy_[ee_index++]*inv_delta_x);
00397                 cc++;
00398             }
00399         }
00400     }
00401 }
00402
00403
00404
00405     for (auto ii = num_extra_rows + 1;
00406         ii < dd_num_rows - num_extra_rows - 1; ii++) {
00407         auto jj = ii - num_extra_rows - 1;
00408         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00409             out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00410         }
00411     }
00412
00413
00414
00415     ee_index = 0;
00416     for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--) {
00417     {
00418         auto cc = 0;
00419         for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00420             if (cc >= elements_per_row) {
00421                 out.SetValue(ii, jj, 0.0);
00422             } else {
00423                 out.SetValue(ii, jj, -mim_bndy_[ee_index++]*inv_delta_x);
00424                 cc++;
00425             }
00426         }
00427     }

```



```

00428
00429     return out;
00430 }
00431
00432 mtk::DenseMatrix mtk::Div1D::ReturnAsDimensionlessDenseMatrix
(
00433     int num_cells_x) const {
00434
00435     int nn{num_cells_x}; // Number of cells on the grid.
00436
00437     #ifdef MTK_PERFORM_PREVENTIONS
00438     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00439     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00440     #endif
00441
00442     int dd_num_rows = nn + 2;
00443     int dd_num_cols = nn + 1;
00444     int elements_per_row = num_bndy_coeffs_;
00445     int num_extra_rows = dim_null_;
00446
00447     // Output matrix featuring sizes for gradient operators.
00448     mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00449
00450
00451     auto ee_index = 0;
00452     for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00453         auto cc = 0;
00454         for (auto jj = 0; jj < dd_num_rows; jj++) {
00455             if (cc >= elements_per_row) {
00456                 out.SetValue(ii, jj, mtk::kZero);
00457             } else {
00458                 out.SetValue(ii, jj, mim_bndy_[ee_index++]);
00459                 cc++;
00460             }
00461         }
00462     }
00463
00464     for (auto ii = num_extra_rows + 1;
00465          ii < dd_num_rows - num_extra_rows - 1; ii++) {
00466         auto jj = ii - num_extra_rows - 1;
00467         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00468             out.SetValue(ii, jj, coeffs_interior_[cc]);
00469         }
00470     }
00471
00472     ee_index = 0;
00473     for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--)
00474     {
00475         auto cc = 0;
00476         for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00477             if (cc >= elements_per_row) {
00478                 out.SetValue(ii, jj, 0.0);
00479             } else {
00480                 out.SetValue(ii, jj, -mim_bndy_[ee_index++]);
00481                 cc++;
00482             }
00483         }
00484     }
00485
00486     return out;
00487 }
00488
00489
00490
00491 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00492
00493     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00494
00495     try {
00496         pp = new mtk::Real[order_accuracy_];
00497     } catch (std::bad_alloc &memory_allocation_exception) {
00498         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00499             std::endl;
00500         std::cerr << memory_allocation_exception.what() << std::endl;
00501     }
00502     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00503
00504     #ifdef MTK_PRECISION_DOUBLE
00505     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00506     #else

```

```

00512 pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00513 #endif
00514
00515 for (auto ii = 1; ii < order_accuracy_; ++ii) {
00516     pp[ii] = pp[ii - 1] + mtk::kOne;
00517 }
00518
00519 #if MTK_VERBOSE_LEVEL > 3
00520 std::cout << "pp =" << std::endl;
00521 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00522     std::cout << std::setw(12) << pp[ii];
00523 }
00524 std::cout << std::endl << std::endl;
00525 #endif
00526
00528
00529 bool transpose{false};
00530
00531 mtk::DenseMatrix vander_matrix(pp,
00532                                 order_accuracy_,
00533                                 order_accuracy_,
00534                                 transpose);
00535
00536 #if MTK_VERBOSE_LEVEL > 4
00537 std::cout << "vander_matrix = " << std::endl;
00538 std::cout << vander_matrix << std::endl;
00539 #endif
00540
00542
00543 try {
00544     coeffs_interior_ = new mtk::Real[order_accuracy_];
00545 } catch (std::bad_alloc &memory_allocation_exception) {
00546     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00547         std::endl;
00548     std::cerr << memory_allocation_exception.what() << std::endl;
00549 }
00550 memset(coeffs_interior_,
00551         mtk::kZero,
00552         sizeof(coeffs_interior_[0])*order_accuracy_);
00553
00554 coeffs_interior_[1] = mtk::kOne;
00555
00556 #if MTK_VERBOSE_LEVEL > 3
00557 std::cout << "oo =" << std::endl;
00558 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00559     std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00560 }
00561 std::cout << std::endl;
00562 #endif
00563
00565
00566 int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00567                                                coeffs_interior_)};
00568
00569 #ifdef MTK_PERFORM_PREVENTIONS
00570 if (!info) {
00571     std::cout << "System solved! Interior stencil attained!" << std::endl;
00572     std::cout << std::endl;
00573 }
00574 else {
00575     std::cerr << "Something wrong solving system! info = " << info << std::endl;
00576     std::cerr << "Exiting..." << std::endl;
00577     return false;
00578 }
00579 #endif
00580
00581 #if MTK_VERBOSE_LEVEL > 3
00582 std::cout << "coeffs_interior_ =" << std::endl;
00583 for (auto ii = 0; ii < order_accuracy_; ++ii) {
00584     std::cout << std::setw(12) << coeffs_interior_[ii];
00585 }
00586 std::cout << std::endl << std::endl;
00587 #endif
00588
00589 delete [] pp;
00590 pp = nullptr;
00591
00592 return true;
00593 }
00594
00595 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {

```

```

00596
00597 mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00598
00600
00601 try {
00602     gg = new mtk::Real[num_bndy_coeffs_];
00603 } catch (std::bad_alloc &memory_allocation_exception) {
00604     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00605         std::endl;
00606     std::cerr << memory_allocation_exception.what() << std::endl;
00607 }
00608 memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00609
00610 #ifdef MTK_PRECISION_DOUBLE
00611 gg[0] = -1.0/2.0;
00612 #else
00613 gg[0] = -1.0f/2.0f;
00614 #endif
00615 for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00616     gg[ii] = gg[ii - 1] + mtk::kOne;
00617 }
00618
00619 #if MTK_VERBOSE_LEVEL > 3
00620 std::cout << "gg =" << std::endl;
00621 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00622     std::cout << std::setw(12) << gg[ii];
00623 }
00624 std::cout << std::endl << std::endl;
00625 #endif
00626
00628
00629 bool tran{true}; // Should I transpose the Vandermonde matrix.
00630
00631 mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00632
00633 #if MTK_VERBOSE_LEVEL > 4
00634 std::cout << "vv_west_t =" << std::endl;
00635 std::cout << vv_west_t << std::endl;
00636 #endif
00637
00639
00640 mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
(vv_west_t));
00641
00642 #if MTK_VERBOSE_LEVEL > 4
00643 std::cout << "QQ^T =" << std::endl;
00644 std::cout << qq_t << std::endl;
00645 #endif
00646
00648
00649 int KK_num_rows_{num_bndy_coeffs_};
00650 int KK_num_cols_{dim_null_};
00651
00652 mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00653
00654 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00655     for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00656         KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00657             qq_t.data()[ii*num_bndy_coeffs_ + jj];
00658     }
00659 }
00660
00661 #if MTK_VERBOSE_LEVEL > 2
00662 std::cout << "KK =" << std::endl;
00663 std::cout << KK << std::endl;
00664 std::cout << "KK.num_rows() =" << KK.num_rows() << std::endl;
00665 std::cout << "KK.num_cols() =" << KK.num_cols() << std::endl;
00666 std::cout << std::endl;
00667 #endif
00668
00670
00671 // Scale thus requesting that the last entries of the attained basis for the
00672 // null-space, adopt the pattern we require.
00673 // Essentially we will implement the following MATLAB pseudo-code:
00674 // scalers = KK(num_bndy_approx - (dim_null - 1):num_bndy_approx,:)\B
00675 // SK = KK*scalers
00676 // where SK is the scaled null-space.
00677
00678 // In this point, we almost have all the data we need correctly allocated
00679 // in memory. We will create the matrix II_, and elements we wish to scale in
00680 // the KK array. Using the concept of the leading dimension, we could just

```

```

00681 // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00682 // GET how does it work. So I will just create a matrix with the content of
00683 // this array that we need, solve for the scalers and then scale the
00684 // whole KK:
00685
00686 // We will then create memory for that sub-matrix of KK (SUBK).
00687
00688 mtk::DenseMatrix SUBK(dim_null_, dim_null_);
00689
00690 for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00691     for (auto jj = 0; jj < dim_null_; ++jj) {
00692         SUBK.data()[ii - (num_bndy_coeffs_ - dim_null_)*dim_null_ + jj] =
00693             KK.data()[ii*dim_null_ + jj];
00694     }
00695 }
00696
00697 #if MTK_VERBOSE_LEVEL > 4
00698 std::cout << "SUBK =" << std::endl;
00699 std::cout << SUBK << std::endl;
00700 #endif
00701
00702 SUBK.Transpose();
00703
00704 #if MTK_VERBOSE_LEVEL > 4
00705 std::cout << "SUBK^T =" << std::endl;
00706 std::cout << SUBK << std::endl;
00707 #endif
00708
00709 bool padded{false};
00710 tran = false;
00711
00712 mtk::DenseMatrix II(dim_null_, padded, tran);
00713
00714 #if MTK_VERBOSE_LEVEL > 4
00715 std::cout << "II =" << std::endl;
00716 std::cout << II << std::endl;
00717 #endif
00718
00719 // Solve the system to compute the scalers.
00720 // An example of the system to solve, for k = 8, is:
00721 //
00722 // SUBK*scalers = II_or
00723 //
00724 // | 0.386018 -0.0339244 -0.129478 |           | 1 0 0 |
00725 // | -0.119774 0.0199423 0.0558632 |*scalers = | 0 1 0 |
00726 // | 0.0155708 -0.00349546 -0.00853182 |       | 0 0 1 |
00727 //
00728 // Notice this is a nrhs = 3 system.
00729 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00730 // will be stored in the created identity matrix.
00731 // Let us first transpose SUBK (because of LAPACK):
00732
00733 int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00734
00735 #ifdef MTK_PERFORM_PREVENTIONS
00736 if (!info) {
00737     std::cout << "System successfully solved!" <<
00738         std::endl;
00739 } else {
00740     std::cerr << "Something went wrong solving system! info = " << info <<
00741         std::endl;
00742     std::cerr << "Exiting..." << std::endl;
00743     return false;
00744 }
00745 std::cout << std::endl;
00746 #endif
00747
00748 #if MTK_VERBOSE_LEVEL > 4
00749 std::cout << "Computed scalers:" << std::endl;
00750 std::cout << II << std::endl;
00751 #endif
00752
00753 // Multiply the two matrices to attain a scaled basis for null-space.
00754
00755 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00756
00757 #if MTK_VERBOSE_LEVEL > 4
00758 std::cout << "Rational basis for the null-space:" << std::endl;
00759 std::cout << rat_basis_null_space_ << std::endl;
00760 #endif
00761

```

```

00762 // At this point, we have a rational basis for the null-space, with the
00763 // pattern we need! :)
00764
00765 delete [] gg;
00766 gg = nullptr;
00767
00768 return true;
00769 }
00770
00771 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00772
00773
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     try {
00805         prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00806     } catch (std::bad_alloc &memory_allocation_exception) {
00807         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00808 std::endl;
00809         std::cerr << memory_allocation_exception.what() << std::endl;
00810     }
00811     memset(prem_apps_,
00812            mtk::kZero,
00813            sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00814
00815     for (auto ll = 0; ll < dim_null_; ++ll) {
00816
00817         // Re-check new generator vector for every iteration except for the first.
00818         #if MTK_VERBOSE_LEVEL > 3
00819         if (ll > 0) {
00820             std::cout << "gg" << ll << " =" << std::endl;
00821             for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00822                 std::cout << std::setw(12) << gg[ii];
00823             }
00824             std::cout << std::endl << std::endl;
00825         }
00826         #endif
00827
00828         bool transpose{false};
00829
00830         mtk::DenseMatrix AA_(gg,
00831                               num_bndy_coeffs_, order_accuracy_ + 1,
00832                               transpose);
00833
00834         #if MTK_VERBOSE_LEVEL > 4
00835         std::cout << "AA_" << ll << " =" << std::endl;
00836         std::cout << AA_ << std::endl;
00837         #endif
00838
00839         mtk::Real *ob{};
00840
00841
00842
00843
00844
00845
00846

```

```

00847     auto ob_ld = num_bndy_coeffs_;
00848
00849     try {
00850         ob = new mtk::Real[ob_ld];
00851     } catch (std::bad_alloc &memory_allocation_exception) {
00852         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00853             std::endl;
00854         std::cerr << memory_allocation_exception.what() << std::endl;
00855     }
00856     memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00857
00858     ob[1] = mtk::kOne;
00859
00860     #if MTK_VERBOSE_LEVEL > 4
00861     std::cout << "ob = " << std::endl << std::endl;
00862     for (auto ii = 0; ii < ob_ld; ++ii) {
00863         std::cout << std::setw(12) << ob[ii] << std::endl;
00864     }
00865     std::cout << std::endl;
00866     #endif
00867
00868     // However, this is an under-determined system of equations. So we can not
00869     // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00870     // our LAPACKAdapter class.
00871
00872     int info_{
00873         mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00874             ob, ob_ld)};
00875
00876     #ifdef MTK_PERFORM_PREVENTIONS
00877     if (!info_) {
00878         std::cout << "System successfully solved!" << std::endl << std::endl;
00879     } else {
00880         std::cerr << "Error solving system! info = " << info_ << std::endl;
00881     }
00882     #endif
00883
00884     #if MTK_VERBOSE_LEVEL > 3
00885     std::cout << "ob =" << std::endl;
00886     for (auto ii = 0; ii < ob_ld; ++ii) {
00887         std::cout << std::setw(12) << ob[ii] << std::endl;
00888     }
00889     std::cout << std::endl;
00890     #endif
00891
00892     // This implies a DAXPY operation. However, we must construct the arguments
00893     // for this operation.
00894
00895     // Save them into the ob_bottom array:
00896
00897     Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00898
00899     try {
00900         ob_bottom = new mtk::Real[dim_null_];
00901     } catch (std::bad_alloc &memory_allocation_exception) {
00902         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00903             std::endl;
00904         std::cerr << memory_allocation_exception.what() << std::endl;
00905     }
00906     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00907
00908     for (auto ii = 0; ii < dim_null_; ++ii) {
00909         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00910     }
00911
00912     #if MTK_VERBOSE_LEVEL > 3
00913     std::cout << "ob_bottom =" << std::endl;
00914     for (auto ii = 0; ii < dim_null_; ++ii) {
00915         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00916     }
00917     std::cout << std::endl;
00918     #endif
00919
00920     // We must computed an scaled ob, sob, using the scaled null-space in
00921     // rat_basis_null_space_.
00922     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00923     // or:
00924     //      ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00925     // thus:
00926     //      Y =      a*A      *x      +      b*Y (DAXPY).

```

```

00931
00932     #if MTK_VERBOSE_LEVEL > 3
00933     std::cout << "Rational basis for the null-space:" << std::endl;
00934     std::cout << rat_basis_null_space_ << std::endl;
00935     #endif
00936
00937     mtk::Real alpha{-mtk::kOne};
00938     mtk::Real beta{mtk::kOne};
00939
00940     mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00941                                   ob_bottom, beta, ob);
00942
00943     #if MTK_VERBOSE_LEVEL > 3
00944     std::cout << "scaled ob:" << std::endl;
00945     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00946         std::cout << std::setw(12) << ob[ii] << std::endl;
00947     }
00948     std::cout << std::endl;
00949     #endif
00950
00951     // We save the recently scaled solution, into an array containing these.
00952     // We can NOT start building the pi matrix, simply because I want that part
00953     // to be separated since its construction depends on the algorithm we want
00954     // to implement.
00955
00956     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00957         prem_apps_[ii*dim_null_ + 11] = ob[ii];
00958     }
00959
00960     // After the first iteration, simply shift the entries of the last
00961     // generator vector used:
00962     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00963         gg[ii]--;
00964     }
00965
00966     // Garbage collection for this loop:
00967     delete[] ob;
00968     ob = nullptr;
00969
00970     delete[] ob_bottom;
00971     ob_bottom = nullptr;
00972 } // End of: for (11 = 0; 11 < dim_null; 11++);
00973
00974 #if MTK_VERBOSE_LEVEL > 4
00975 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00976 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00977     for (auto jj = 0; jj < dim_null_; ++jj) {
00978         std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00979     }
00980     std::cout << std::endl;
00981 }
00982 std::cout << std::endl;
00983 #endif
00984
00985 delete[] gg;
00986 gg = nullptr;
00987
00988 return true;
00989 }
00990
00991 bool mtk::Div1D::ComputeWeights(void) {
00992
00993     // Matrix to compute the weights as in the CRSA.
00994     mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00995
00996
00997     // Assemble the pi matrix using:
00998     // 1. The collection of scaled preliminary approximations.
00999     // 2. The collection of coefficients approximating at the interior.
01000     // 3. The scaled basis for the null-space.
01001
01002     // 1.1. Process array of scaled preliminary approximations.
01003
01004     // These are queued in scaled_solutions. Each one of these, will be a column
01005     // of the pi matrix:
01006     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01007         for (auto jj = 0; jj < dim_null_; ++jj) {
01008             pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01009                 prem_apps_[ii*dim_null_ + jj];
01010         }
01011     }
01012 }

```

```

01013
01014 // 1.2. Add columns from known stencil approximating at the interior.
01015
01016 // However, these must be padded by zeros, according to their position in the
01017 // final pi matrix:
01018 auto mm = 0;
01019 for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
01020     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01021         pi.data()[ (ii + mm)*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
01022             coeffs_interior_[ii];
01023     }
01024     ++mm;
01025 }
01026
01027 rat_basis_null_space_.OrderColMajor();
01028
01029 #if MTK_VERBOSE_LEVEL > 4
01030 std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01031 std::cout << rat_basis_null_space_ << std::endl;
01032 #endif
01033
01034 // 1.3. Add final set of columns: rational basis for null-space.
01035 for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01036     jj < num_bndy_coeffs_ - 1;
01037     ++jj) {
01038     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01039         auto og =
01040             (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01041         auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01042         pi.data()[de] = rat_basis_null_space_.data()[og];
01043     }
01044 }
01045
01046 #if MTK_VERBOSE_LEVEL > 3
01047 std::cout << "coeffs_interior_ =" << std::endl;
01048 for (auto ii = 0; ii < order_accuracy_; ++ii) {
01049     std::cout << std::setw(12) << coeffs_interior_[ii];
01050 }
01051 std::cout << std::endl << std::endl;
01052 #endif
01053
01054 #if MTK_VERBOSE_LEVEL > 4
01055 std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01056 std::cout << pi << std::endl;
01057 #endif
01058
01060 // This imposes the mimetic condition.
01061
01062 mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01063
01064 try {
01065     hh = new mtk::Real[num_bndy_coeffs_];
01066 } catch (std::bad_alloc &memory_allocation_exception) {
01067     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01068         std::endl;
01069     std::cerr << memory_allocation_exception.what() << std::endl;
01070 }
01071
01072 memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01073
01074 hh[0] = -mtk::kOne;
01075 for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01076     auto aux_xx = mtk::kZero;
01077     for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01078         aux_xx += coeffs_interior_[jj];
01079     }
01080     hh[ii] = -mtk::kOne*aux_xx;
01081 }
01082
01084 // That is, we construct a system, to solve for the weights.
01085
01086 // Once again we face the challenge of solving with LAPACK. However, for the
01087 // CRS, this matrix PI is over-determined, since it has more rows than
01088 // unknowns. However, according to the theory, the solution to this system is
01089 // unique. We will use dgels_.
01090
01091 try {
01092     weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01093 } catch (std::bad_alloc &memory_allocation_exception) {
01094     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```



```

01096         std::endl;
01097         std::cerr << memory_allocation_exception.what() << std::endl;
01098     }
01099     memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01100
01101     int weights_ld{pi.num_cols() + 1};
01102
01103     // Preserve hh.
01104     std::copy(hh, hh + weights_ld, weights_cbs_);
01105
01106     pi.Transpose();
01107
01108     int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
01109         pi,
01110         weights_cbs_,
01111         weights_ld)};
01112
01113     #ifdef MTK_PERFORM_PREVENTIONS
01114     if (!info) {
01115         std::cout << "System successfully solved!" << std::endl << std::endl;
01116     } else {
01117         std::cerr << "Error solving system! info = " << info << std::endl;
01118     }
01119     #endif
01120
01121     #if MTK_VERBOSE_LEVEL > 3
01122     std::cout << "hh =" << std::endl;
01123     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01124         std::cout << std::setw(11) << hh[ii] << std::endl;
01125     }
01126     std::cout << std::endl;
01127     #endif
01128     // Preserve the original weights for research.
01129
01130     try {
01131         weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01132     } catch (std::bad_alloc &memory_allocation_exception) {
01133         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01134             std::endl;
01135         std::cerr << memory_allocation_exception.what() << std::endl;
01136     }
01137     memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01138
01139     std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01140
01141     #if MTK_VERBOSE_LEVEL > 3
01142     std::cout << "weights_CRSA + lambda =" << std::endl;
01143     for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01144         std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01145     }
01146     std::cout << std::endl;
01147     #endif
01148
01149     if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01150
01151         mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01152
01153         for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01154             for (auto jj = 0; jj < dim_null_; ++jj) {
01155                 phi.data()[ii*(order_accuracy_ + 1) + jj] = prem_apps_[ii*dim_null_ + jj];
01156             }
01157         }
01158
01159         int aux{}; // Auxiliary variable.
01160         for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01161             for (auto ii = 0; ii < order_accuracy_; ++ii) {
01162                 phi.data()[ii*(order_accuracy_ + 1) + jj] = coeffs_interior[ii];
01163             }
01164             ++aux;
01165         }
01166
01167         for (auto jj = order_accuracy_ - 1; jj >= order_accuracy_ - dim_null_; jj--) {
01168             for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01169                 phi.data()[ii*(order_accuracy_ + 1) + jj] = mtk::kZero;
01170             }
01171         }
01172
01173         for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {

```

```

01178     for (auto ii = 0; ii < dim_null_; ++ii) {
01179         phi.data()[ (ii + order_accuracy_ - dim_null_ + jj*order_accuracy_) ] =
01180             -prem_apps_[ (dim_null_ - ii - 1 + jj*dim_null_) ];
01181     }
01182 }
01183
01184 for (auto ii = 0; ii < order_accuracy_/2; ++ii) {
01185     for (auto jj = dim_null_ + 2; jj < order_accuracy_; ++jj) {
01186         auto swap = phi.data()[ ii*order_accuracy_+jj ];
01187         phi.data()[ ii*order_accuracy_ + jj ] =
01188             phi.data()[ (order_accuracy_-ii)*order_accuracy_+jj ];
01189         phi.data()[ (order_accuracy_-ii)*order_accuracy_+jj ] = swap;
01190     }
01191 }
01192
01193 #if MTK_VERBOSE_LEVEL > 4
01194 std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01195 std::cout << phi << std::endl;
01196 #endif
01197
01198 mtk::Real *lamed{}; // Used to build big lambda.
01199
01200 try {
01201     lamed = new mtk::Real[dim_null_];
01202 } catch (std::bad_alloc &memory_allocation_exception) {
01203     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01204         std::endl;
01205     std::cerr << memory_allocation_exception.what() << std::endl;
01206 }
01207 memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01208
01209 for (auto ii = 0; ii < dim_null_; ++ii) {
01210     lamed[ii] = hh[ii + order_accuracy_ + 1];
01211 }
01212
01213 #if MTK_VERBOSE_LEVEL > 3
01214 std::cout << "lamed =" << std::endl;
01215 for (auto ii = 0; ii < dim_null_; ++ii) {
01216     std::cout << std::setw(12) << lamed[ii] << std::endl;
01217 }
01218 std::cout << std::endl;
01219 #endif
01220
01221 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01222     mtk::Real temp = mtk::kZero;
01223     for (auto jj = 0; jj < dim_null_; ++jj) {
01224         temp = temp +
01225             lamed[jj]*rat_basis_null_space_.data()[ jj*num_bndy_coeffs_ + ii ];
01226     }
01227     hh[ii] = hh[ii] - temp;
01228 }
01229
01230 #if MTK_VERBOSE_LEVEL > 3
01231 std::cout << "big_lambda =" << std::endl;
01232 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01233     std::cout << std::setw(12) << hh[ii] << std::endl;
01234 }
01235 std::cout << std::endl;
01236 #endif
01237
01238 #ifdef MTK_VERBOSE_WEIGHTS
01239 int copy_result{1};
01240 #else
01241 int copy_result{};
01242 #endif
01243
01244 mtk::Real normerr_; // Norm of the error for the solution on each row.
01245
01246 int minrow_{std::numeric_limits<int>::infinity()};
01247
01248 mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_crs_,
01249     order_accuracy_)};
01250 mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01251
01252 #ifdef MTK_VERBOSE_WEIGHTS
01253 std::ofstream table("div_ld_" + std::to_string(order_accuracy_) +
01254     "_weights.tex");
01255 table << "\\begin{tabular}{c}{c}";

```

```

01260     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01261         table << 'c';
01262     }
01263     table << ":c)\n\\toprule\nRow & ";
01264     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01265         table << "$ q_{ " + std::to_string(ii) + " }$ & ";
01266     }
01267     table << " Relative error \\\n\\midrule\n";
01268     #endif
01269
01270     for(auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01271         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01272                                     order_accuracy_ + 1,
01273                                     order_accuracy_,
01274                                     order_accuracy_,
01275                                     hh,
01276                                     weights_cbs_,
01277                                     row_,
01278                                     mimetic_threshold_,
01279                                     copy_result);
01280         mtk::Real aux{normerr_/norm_};
01281
01282         #if MTK_VERBOSE_LEVEL > 2
01283         std::cout << "Relative norm: " << aux << " " << std::endl;
01284         std::cout << std::endl;
01285         #endif
01286
01287         if (aux < minnorm_) {
01288             minnorm_ = aux;
01289             minrow_ = row_;
01290         }
01291
01292         #ifdef MTK_VERBOSE_WEIGHTS
01293         table << std::to_string(row_ + 1) << " & ";
01294         if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01295             for (int ii = 1; ii <= order_accuracy_; ++ii) {
01296                 table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01297             }
01298             table << std::to_string(aux) << " \\\n" << std::endl;
01299         } else {
01300             table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
01301                 "}{c}{\\emptyset$} & ";
01302             table << " - \\\n" << std::endl;
01303         }
01304         #endif
01305     }
01306
01307     #ifdef MTK_VERBOSE_WEIGHTS
01308     table << "\\midrule" << std::endl;
01309     table << "CRS weights:";
01310     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01311         table << " & " << std::to_string(weights_crs_[ii - 1]);
01312     }
01313     table << " & - \\\n\\bottomrule\n\\end{tabular}" << std::endl;
01314     table.close();
01315     #endif
01316
01317     #if MTK_VERBOSE_LEVEL > 3
01318     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01319         std::endl;
01320     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01321         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01322     }
01323     std::cout << std::endl;
01324     #endif
01325
01326     // After we know which row yields the smallest relative norm that row is
01327     // chosen to be the objective function and the result of the optimizer is
01328     // chosen to be the new weights_.
01329
01330     #if MTK_VERBOSE_LEVEL > 2
01331     std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01332         minrow_ + 1 << std::endl;
01333     std::cout << std::endl;
01334     #endif
01335
01336     copy_result = 1;
01337     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),

```

```

01340                                     order_accuracy_ + 1,
01341                                     order_accuracy_,
01342                                     order_accuracy_,
01343                                     hh,
01344                                     weights_cbs_,
01345                                     minrow_,
01346                                     mimetic_threshold_,
01347                                     copy_result);
01348     mtk::Real aux_{normerr_/norm_};
01349     #if MTK_VERBOSE_LEVEL > 2
01350     std::cout << "Relative norm: " << aux_ << std::endl;
01351     std::cout << std::endl;
01352     #endif
01353
01354     delete [] lamed;
01355     lamed = nullptr;
01356 }
01357
01358 delete [] hh;
01359 hh = nullptr;
01360
01361 return true;
01362 }
01363
01364 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01365
01366     #if MTK_VERBOSE_LEVEL > 3
01367     std::cout << "weights_CBSA + lambda =" << std::endl;
01368     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01369         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01370     }
01371     std::cout << std::endl;
01372     #endif
01373
01374     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01375
01376     try {
01377         lambda = new mtk::Real[dim_null_];
01378     } catch (std::bad_alloc &memory_allocation_exception) {
01379         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01380             std::endl;
01381         std::cerr << memory_allocation_exception.what() << std::endl;
01382     }
01383     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01384
01385     for (auto ii = 0; ii < dim_null_; ++ii) {
01386         lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01387     }
01388
01389     #if MTK_VERBOSE_LEVEL > 3
01390     std::cout << "lambda =" << std::endl;
01391     for (auto ii = 0; ii < dim_null_; ++ii) {
01392         std::cout << std::setw(12) << lambda[ii] << std::endl;
01393     }
01394     std::cout << std::endl;
01395     #endif
01396
01397     mtk::Real *alpha{}; // Collection of alpha values.
01398
01399     try {
01400         alpha = new mtk::Real[dim_null_];
01401     } catch (std::bad_alloc &memory_allocation_exception) {
01402         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01403             std::endl;
01404         std::cerr << memory_allocation_exception.what() << std::endl;
01405     }
01406     memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01407
01408     for (auto ii = 0; ii < dim_null_; ++ii) {
01409         alpha[ii] = lambda[ii]/weights_cbs_[ii];
01410     }
01411
01412     #if MTK_VERBOSE_LEVEL > 3
01413     std::cout << "alpha =" << std::endl;
01414     for (auto ii = 0; ii < dim_null_; ++ii) {
01415         std::cout << std::setw(12) << alpha[ii] << std::endl;
01416     }
01417     std::cout << std::endl;
01418     #endif
01419 }

```

```

01423
01425
01426     try {
01427         mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01428     } catch (std::bad_alloc &memory_allocation_exception) {
01429         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01430             std::endl;
01431         std::cerr << memory_allocation_exception.what() << std::endl;
01432     }
01433     memset(mim_bndy_,
01434         mtk::kZero,
01435         sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01436
01437     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01438         for (auto jj = 0; jj < dim_null_; ++jj) {
01439             mim_bndy_[ii*dim_null_ + jj] =
01440                 prem_apps_[ii*dim_null_ + jj] +
01441                 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01442         }
01443     }
01444
01445     #if MTK_VERBOSE_LEVEL > 3
01446     std::cout << "Collection of mimetic approximations:" << std::endl;
01447     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01448         for (auto jj = 0; jj < dim_null_; ++jj) {
01449             std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01450         }
01451         std::cout << std::endl;
01452     }
01453     std::cout << std::endl;
01454     #endif
01455
01456     for (auto ii = 0; ii < dim_null_; ++ii) {
01457         sums_rows_mim_bndy_.push_back(mtk::kZero);
01458         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01459             sums_rows_mim_bndy_[ii] += mim_bndy_[jj*dim_null_ + ii];
01460         }
01461     }
01462
01463     #if MTK_VERBOSE_LEVEL > 3
01464     std::cout << "Row-wise sum of mimetic approximations:" << std::endl;
01465     for (auto ii = 0; ii < dim_null_; ++ii) {
01466         std::cout << std::setw(13) << sums_rows_mim_bndy_[ii];
01467     }
01468     std::cout << std::endl;
01469     std::cout << std::endl;
01470     #endif
01471     delete[] lambda;
01472     lambda = nullptr;
01473
01474     delete[] alpha;
01475     alpha = nullptr;
01476
01477     return true;
01478 }
01479
01480 bool mtk::Div1D::AssembleOperator(void) {
01481
01482     // The output array will have this form:
01483     // 1. The first entry of the array will contain used order order_accuracy_.
01484     // 2. The second entry of the array will contain the collection of
01485     // approximating coefficients for the interior of the grid.
01486     // 3. IF order_accuracy_ > 2, then the third entry will contain a collection
01487     // of weights.
01488     // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the
01489     // collections of approximating coefficients for the west boundary of the
01490     // grid.
01491
01492     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01493         divergence_length_ =
01494             1 + order_accuracy_ + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01495     } else {
01496         divergence_length_ = 1 + order_accuracy_;
01497     }
01498
01499     #if MTK_VERBOSE_LEVEL > 2
01500     std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01501     std::cout << std::endl;
01502     #endif

```

```

01506
01507     try {
01508         divergence_ = new double[divergence_length_];
01509     } catch (std::bad_alloc &memory_allocation_exception) {
01510         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01511             std::endl;
01512         std::cerr << memory_allocation_exception.what() << std::endl;
01513     }
01514     memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01515
01516     divergence_[0] = order_accuracy_;
01517
01518     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01519         divergence_[ii + 1] = coeffs_interior_[ii];
01520     }
01521
01522     if (order_accuracy_ > 2) {
01523         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01524             divergence_[1 + order_accuracy_ + ii] = weights_cbs_[ii];
01525         }
01526     }
01527
01528     if (order_accuracy_ > 2) {
01529         auto offset = (2*order_accuracy_ + 1);
01530         int mm{};
01531         for (auto ii = 0; ii < dim_null_; ++ii) {
01532             for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01533                 divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01534                 ++mm;
01535             }
01536         }
01537     }
01538
01539     #if MTK_VERBOSE_LEVEL > 1
01540     std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01541     std::cout << std::endl;
01542     #endif
01543     return true;
01544 }

```

18.87 src/mtk_div_2d.cc File Reference

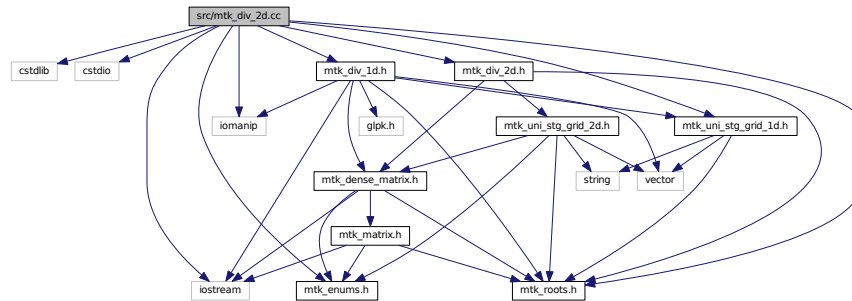
Implements the class Div2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"

```

Include dependency graph for mtk_div_2d.cc:



18.87.1 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](#).

18.88 mtk_div_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044

```

```

00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_enums.h"
00065 #include "mtk_uni_stg_grid_ld.h"
00066 #include "mtk_div_ld.h"
00067 #include "mtk_div_2d.h"
00068
00069 mtk::Div2D::Div2D():
00070     order_accuracy_(),
00071     mimetic_threshold_() {}
00072
00073 mtk::Div2D::Div2D(const Div2D &div):
00074     order_accuracy_(div.order_accuracy_),
00075     mimetic_threshold_(div.mimetic_threshold_) {}
00076
00077 mtk::Div2D::~Div2D() {}
00078
00079 bool mtk::Div2D::ConstructDiv2D(const
    mtk::UniStgGrid2D &grid,
                                int order_accuracy,
                                mtk::Real mimetic_threshold) {
00080
00081
00082
00083     int num_cells_x = grid.num_cells_x();
00084     int num_cells_y = grid.num_cells_y();
00085
00086     int mx = num_cells_x + 2; // Dx vertical dimension.
00087     int nx = num_cells_x + 1; // Dx horizontal dimension.
00088     int my = num_cells_y + 2; // Dy vertical dimension.
00089     int ny = num_cells_y + 1; // Dy horizontal dimension.
00090
00091     mtk::Div1D div;
00092
00093     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00094
00095     #ifdef MTK_PERFORM_PREVENTIONS
00096     if (!info) {
00097         std::cerr << "Mimetic div could not be built." << std::endl;
00098         return info;
00099     }
00100     #endif
00101
00102     auto west = grid.west_bndy();
00103     auto east = grid.east_bndy();
00104     auto south = grid.south_bndy();
00105     auto north = grid.east_bndy();
00106
00107     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00108     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00109
00110     mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00111     mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00112
00113     bool padded{true};
00114     bool transpose{false};
00115
00116     mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00117     mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00118
00119     mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00120     mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00121
00122     #if MTK_VERBOSE_LEVEL > 2
00123     std::cout << "Dx: " << mx << " by " << nx << std::endl;
00124     std::cout << "Iy : " << num_cells_y << " by " << ny << std::endl;

```



```

00125     std::cout << "Dy: " << my << " by " << ny << std::endl;
00126     std::cout << "Ix : " << num_cells_x << " by " << nx << std::endl;
00127     std::cout << "Div 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00128         nx*ny << std::endl;
00129     #endif
00130
00131     mtk::DenseMatrix d2d(mx*my, nx*num_cells_y + ny*num_cells_x);
00132
00133     for (auto ii = 0; ii < mx*my; ii++) {
00134         for (auto jj = 0; jj < nx*num_cells_y; jj++) {
00135             d2d.SetValue(ii, jj, dxy.GetValue(ii, jj));
00136         }
00137         for (auto kk = 0; kk < ny*num_cells_x; kk++) {
00138             d2d.SetValue(ii, kk + nx*num_cells_y, dyx.GetValue(ii, kk));
00139         }
00140     }
00141
00142     divergence_ = d2d;
00143
00144     return info;
00145 }
00146
00147 mtk::DenseMatrix mtk::Div2D::ReturnAsDenseMatrix() const {
00148
00149     return divergence_;
00150 }

```

18.89 src/mtk_div_3d.cc File Reference

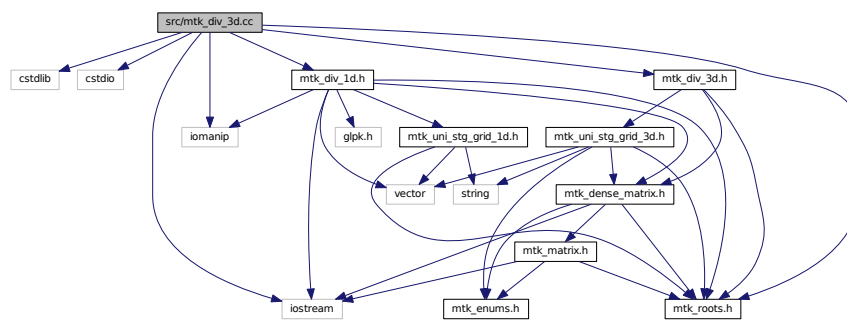
Implements the class Div3D.

```

#include <cstdlib>
#include <stdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_div_1d.h"
#include "mtk_div_3d.h"

```

Include dependency graph for mtk_div_3d.cc:



18.89.1 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](#).

18.90 mtk_div_3d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_div_1d.h"
00065 #include "mtk_div_3d.h"
00066
00067 mtk::Div3D::Div3D():
00068     order_accuracy_(1),
00069     mimetic_threshold_(1) {}
00070
00071 mtk::Div3D::Div3D(const Div3D &grad):
00072     order_accuracy_(grad.order_accuracy_),
00073     mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Div3D::~Div3D() {}
00076
00077 bool mtk::Div3D::ConstructDiv3D(const

```

```

    mtk::UniStgGrid3D &grid,
00078                                     int order_accuracy,
00079                                     mtk::Real mimetic_threshold) {
00080
00081     int num_cells_x = grid.num_cells_x();
00082     int num_cells_y = grid.num_cells_y();
00083     int num_cells_z = grid.num_cells_z();
00084
00085     int mx = num_cells_x + 1; // Dx vertical dimension.
00086     int nx = num_cells_x + 2; // Dx horizontal dimension.
00087     int my = num_cells_y + 1; // Dy vertical dimension.
00088     int ny = num_cells_y + 2; // Dy horizontal dimension.
00089     int mz = num_cells_z + 1; // Dz vertical dimension.
00090     int nz = num_cells_z + 2; // Dz horizontal dimension.
00091
00092     mtk::Div1D div;
00093
00094     bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00095
00096     #ifdef MTK_PERFORM_PREVENTIONS
00097     if (!info) {
00098         std::cerr << "Mimetic div could not be built." << std::endl;
00099         return info;
00100     }
00101     #endif
00102
00103     auto west = grid.west_bndy();
00104     auto east = grid.east_bndy();
00105     auto south = grid.south_bndy();
00106     auto north = grid.east_bndy();
00107     auto bottom = grid.bottom_bndy();
00108     auto top = grid.top_bndy();
00109
00110     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00111     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00112     mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z);
00113
00114     mtk::DenseMatrix Dx(div.ReturnAsDenseMatrix(grid_x));
00115     mtk::DenseMatrix Dy(div.ReturnAsDenseMatrix(grid_y));
00116     mtk::DenseMatrix Dz(div.ReturnAsDenseMatrix(grid_z));
00117
00118     bool padded{true};
00119     bool transpose{false};
00120
00121     mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00122     mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00123     mtk::DenseMatrix iz(num_cells_z, padded, transpose);
00124
00125
00126
00127     mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(iz, iy));
00128     mtk::DenseMatrix dx(mtk::DenseMatrix::Kron(aux1, Dx));
00129
00130
00131
00132     mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(iz, Dy));
00133     mtk::DenseMatrix dy(mtk::DenseMatrix::Kron(aux2, ix));
00134
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 }

```

18.91 src/mtk_glpk_adapter.cc File Reference

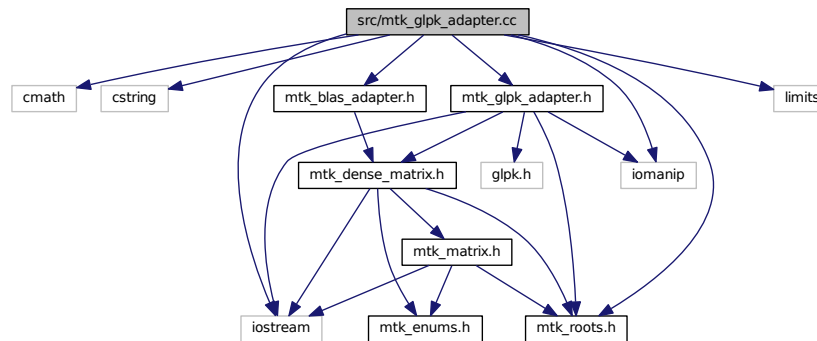
Adapter class for the GLPK API.

```

#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_glpk_adapter.h"

```

Include dependency graph for mtk_glpk_adapter.cc:



18.91.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the GLPK.

The **GLPK (GNU Linear Programming Kit)** package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

See also

<http://www.gnu.org/software/glpk/>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_glpk_adapter.cc](#).

18.92 mtk_glpk_adapter.cc

```

00001
00020 /*
00021 Copyright (C) 2015, Computational Science Research Center, San Diego State
00022 University. All rights reserved.
00023
00024 Redistribution and use in source and binary forms, with or without modification,
00025 are permitted provided that the following conditions are met:
00026
00027 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00028 and a copy of the modified files should be reported once modifications are
00029 completed, unless these modifications are made through the project's GitHub
00030 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00031 should be developed and included in any deliverable.
00032
00033 2. Redistributions of source code must be done through direct
00034 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00035
00036 3. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 4. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders, and due credit should
00042 be given to the copyright holders.
00043
00044 5. Neither the name of the copyright holder nor the names of its contributors
00045 may be used to endorse or promote products derived from this software without
00046 specific prior written permission.
00047
00048 The copyright holders provide no reassurances that the source code provided does
00049 not infringe any patent, copyright, or any other intellectual property rights of
00050 third parties. The copyright holders disclaim any liability to any recipient for
00051 claims brought against recipient by any third party for infringement of that
00052 parties intellectual property rights.
00053
00054 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00055 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00056 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00057 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00058 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00059 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00060 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00061 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00062 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00063 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00064 */
00065

```

```

00066 #include <cmath>
00067 #include <cstring>
00068
00069 #include <iostream>
00070 #include <iomanip>
00071 #include <limits>
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_blas_adapter.h"
00075 #include "mtk_glpk_adapter.h"
00076
00077 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
    mtk::Real *A,
00078
00079         int nrows,
00080         int ncols,
00081         int kk,
00082         mtk::Real *hh,
00083         mtk::Real *qq,
00084         int robjective,
00085         mtk::Real mimetic_threshold,
00086         int copy) {
00087
00088     #if MTK_DEBUG_LEVEL > 0
00089     char mps_file_name[18]; // File name for the MPS files.
00090     #endif
00091     char rname[5];          // Row name.
00092     char cname[5];          // Column name.
00093
00094     glp_prob *lp; // Linear programming problem.
00095
00096     int *ia; // Array for the problem.
00097     int *ja; // Array for the problem.
00098
00099     int problem_size; // Size of the problem.
00100     int lp_nrows;     // Number of rows.
00101     int lp_ncols;     // Number of columns.
00102     int matsize;      // Size of the matrix.
00103     int glp_index{1}; // Index of the objective function.
00104     int ii;            // Iterator.
00105     int jj;            // Iterator.
00106
00107     mtk::Real *ar;      // Array for the problem.
00108     mtk::Real *objective; // Array containing the objective function.
00109     mtk::Real *rhs;      // Array containing the rhs.
00110     mtk::Real *err;      // Array of errors.
00111
00112     mtk::Real x1;        // Norm-2 of the error.
00113
00114     #if MTK_DEBUG_LEVEL > 0
00115     mtk::Real obj_value; // Value of the objective function.
00116     #endif
00117
00118     lp_nrows = kk;
00119     lp_ncols = kk;
00120
00121     matsize = lp_nrows*lp_ncols;
00122
00123
00124     problem_size = lp_nrows*lp_ncols + 1;
00125
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     memset(ia, 0, sizeof(ia[0])*problem_size);
00135
00136     try {
00137         ja = new int[problem_size];
00138     } catch (std::bad_alloc &memory_allocation_exception) {
00139         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00140             std::endl;
00141         std::cerr << memory_allocation_exception.what() << std::endl;
00142     }
00143     memset(ja, 0, sizeof(ja[0])*problem_size);
00144
00145     try {
00146         ar = new mtk::Real[problem_size];
00147     } catch (std::bad_alloc &memory_allocation_exception) {

```

```

00148     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00149     std::endl;
00150     std::cerr << memory_allocation_exception.what() << std::endl;
00151 }
00152 memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00153
00154 try {
00155     objective = new mtk::Real[lp_ncols + 1];
00156 } catch (std::bad_alloc &memory_allocation_exception) {
00157     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00158     std::endl;
00159     std::cerr << memory_allocation_exception.what() << std::endl;
00160 }
00161 memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00162
00163 try {
00164     rhs = new mtk::Real[lp_nrows + 1];
00165 } catch (std::bad_alloc &memory_allocation_exception) {
00166     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00167     std::endl;
00168     std::cerr << memory_allocation_exception.what() << std::endl;
00169 }
00170 memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00171
00172 try {
00173     err = new mtk::Real[lp_nrows];
00174 } catch (std::bad_alloc &memory_allocation_exception) {
00175     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00176     std::endl;
00177     std::cerr << memory_allocation_exception.what() << std::endl;
00178 }
00179 memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00180
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;
00186 #endif
00187
00188 lp = glp_create_prob();
00189
00190 glp_set_prob_name (lp, "mtk:GLPKAdapter::Simplex");
00191
00192 glp_set_obj_dir (lp, GLP_MIN);
00193
00194 glp_add_rows(lp, lp_nrows);
00195
00196 for (ii = 1; ii <= lp_nrows; ++ii) {
00197     sprintf(rname, "R%02d",ii);
00198     glp_set_row_name(lp, ii, rname);
00199 }
00200
00201 glp_add_cols(lp, lp_ncols);
00202
00203 for (ii = 1; ii <= lp_ncols; ++ii) {
00204     sprintf(cname, "Q%02d",ii);
00205     glp_set_col_name (lp, ii, cname);
00206 }
00207
00208 #if MTK_DEBUG_LEVEL>0
00209 std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
00210 #endif
00211
00212 for (jj = 0; jj < kk; ++jj) {
00213     objective[glp_index] = A[jj + robjective * ncols];
00214     glp_index++;
00215 }
00216
00217 #if MTK_DEBUG_LEVEL > 0
00218 std::cout << std::endl;
00219 #endif
00220
00221 glp_index = 1;
00222 rhs[0] = mtk::kZero;
00223 for (ii = 0; ii <= lp_nrows; ++ii) {
00224     if (ii != robjective) {
00225         rhs[glp_index] = hh[ii];
00226         glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00227         glp_index++;
00228     }
00229 }

```

```

00232     }
00233 }
00234
00235 #if MTK_DEBUG_LEVEL > 0
00236 std::cout << "rhs =" << std::endl;
00237 for (auto ii = 0; ii < lp_nrows; ++ii) {
00238     std::cout << std::setw(15) << rhs[ii] << std::endl;
00239 }
00240 std::cout << std::endl;
00241 #endif
00242
00243
00244
00245 for (ii = 1; ii <= lp_ncols; ++ii) {
00246     glp_set_obj_coef (lp, ii, objective[ii]);
00247 }
00248
00249
00250
00251 for (ii = 1; ii <= lp_ncols; ++ii) {
00252     glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00253 }
00254
00255
00256
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 glp_index = 0;
00268
00269 for (ii = 1; ii < problem_size; ++ii) {
00270     if ((ii - 1) % lp_ncols == 0) {
00271         glp_index++;
00272     }
00273     ia[ii] = glp_index;
00274     ja[ii] = (ii - 1) % lp_ncols + 1;
00275 }
00276
00277 glp_load_matrix (lp, matsize, ia, ja, ar);
00278
00279 #if MTK_DEBUG_LEVEL > 0
00280 sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00281 glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00282 #endif
00283
00284
00285
00286 glp_simplex (lp, nullptr);
00287
00288 // Check status of the solution.
00289
00290 if (glp_get_status(lp) == GLP_OPT) {
00291
00292     for(ii = 1; ii <= lp_ncols; ++ii) {
00293         err[ii - 1] = qq[ii - 1] - glp_get_col_prim(lp, ii);
00294     }
00295
00296     #if MTK_DEBUG_LEVEL > 0
00297     obj_value = glp_get_obj_val (lp);
00298     std::cout << std::setw(12) << "CRS" << std::setw(12) << "CRS" << std::endl;
00299     for (ii = 0; ii < lp_ncols; ++ii) {
00300         std::cout << "q_" << ii + 1 << " = " << std::setw(12) <<
00301             glp_get_col_prim(lp, ii + 1) << std::setw(12) << qq[ii] << std::endl;
00302     }
00303     std::cout << "Objective function value (row " << robjective + 1 << ") = " <<
00304         obj_value << std::endl;
00305     #endif
00306
00307     if (copy) {
00308         for(ii = 0; ii < lp_ncols; ++ii) {
00309             qq[ii] = glp_get_col_prim(lp, ii + 1);
00310         }
00311         // Preserve the bottom values of qq.
00312     }
00313
00314     x1 = mtk::BLASAdapter::RealNRM2(err, lp_ncols);
00315
00316 } else {

```



```

00317     x1 = std::numeric_limits<mtk::Real>::infinity();
00318 }
00319
00320 glp_delete_prob (lp);
00321 glp_free_env ();
00322
00323 delete [] ia;
00324 delete [] ja;
00325 delete [] ar;
00326 delete [] objective;
00327 delete [] rhs;
00328 delete [] err;
00329
00330 return x1;
00331 }

```

18.93 src/mtk_grad_1d.cc File Reference

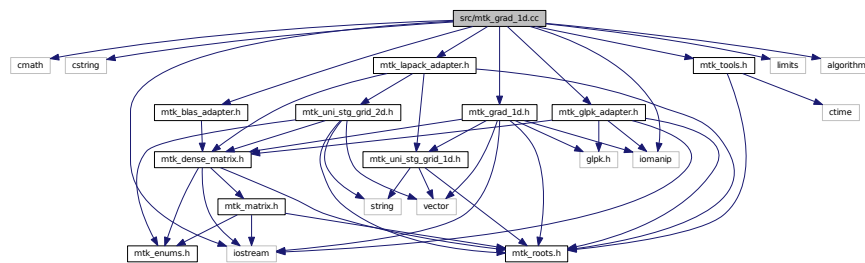
Implements the class Grad1D.

```

#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)`

18.93.1 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](#).

18.94 mtk_grad_1d.cc

```

00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066

```

```

00067 #ifdef MTK_VERBOSE_WEIGHTS
00068 #include <fstream>
00069 #endif
00070
00071 #include <limits>
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075
00076 #include "mtk_blas_adapter.h"
00077 #include "mtk_lapack_adapter.h"
00078 #include "mtk_glpk_adapter.h"
00079
00080 #include "mtk_grad_1d.h"
00081
00082 namespace mtk {
00083
00084 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &in) {
00085     int output_precision{4};
00086     int output_width{8};
00087
00088     stream << "Order of accuracy: " << in.gradient_[0] << std::endl;
00089
00090     stream << "Interior stencil: " << std::endl;
00091     for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00092         stream << std::setprecision(output_precision) <<
00093             std::setw(output_width) << in.gradient_[ii] << ' ';
00094     }
00095     stream << std::endl;
00096
00097     stream << "Weights:" << std::endl;
00098     for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00099 order_accuracy_; ++ii) {
00100         stream << std::setprecision(output_precision) <<
00101             std::setw(output_width) << in.gradient_[ii] << ' ';
00102     }
00103     stream << std::endl;
00104
00105     int offset{2*in.order_accuracy_ + 1};
00106     int mm {};
00107     if (in.order_accuracy_ > mtk::kDefaultOrderAccuracy) {
00108         for (auto ii = 0; ii < in.num_bndy_approxs_; ++ii) {
00109             stream << "Mimetic boundary row " << ii + 1 << ": " << std::endl;
00110             for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {
00111                 auto value = in.gradient_[offset + (mm)];
00112                 stream << std::setprecision(output_precision) <<
00113                     std::setw(output_width) << value << ' ';
00114                 mm++;
00115             }
00116             stream << std::endl;
00117             stream << "Sum of elements in row " << ii + 1 << ": " <<
00118                 in.sums_rows_mim_bndy[ii];
00119             stream << std::endl;
00120         }
00121     } else {
00122         stream << std::setprecision(output_precision) <<
00123             std::setw(output_width) << in.gradient_[offset + 0] << ' ';
00124         stream << std::setprecision(output_precision) <<
00125             std::setw(output_width) << in.gradient_[offset + 1] << ' ';
00126         stream << std::setprecision(output_precision) <<
00127             std::setw(output_width) << in.gradient_[offset + 2] << ' ';
00128         stream << std::endl;
00129     }
00130     return stream;
00131 }
00132
00133 mtk::Grad1D::Grad1D():
00134     order_accuracy_(mtk::kDefaultOrderAccuracy),
00135     dim_null_(),
00136     num_bndy_approxs_(),
00137     num_bndy_coeffs_(),
00138     gradient_length_(),
00139     minrow_(),
00140     row_(),

```

```

00151     coeffs_interior_(),
00152     prem_apps_(),
00153     weights_crs_(),
00154     weights_cbs_(),
00155     mim_bndy_(),
00156     gradient_(),
00157     sums_rows_mim_bndy_(),
00158     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00159
00160 mtk::Grad1D::Grad1D(const Grad1D &grad):
00161     order_accuracy_(grad.order_accuracy_),
00162     dim_null_(grad.dim_null_),
00163     num_bndy_approxs_(grad.num_bndy_approxs_),
00164     num_bndy_coeffs_(grad.num_bndy_coeffs_),
00165     gradient_length_(grad.gradient_length_),
00166     minrow_(grad.minrow_),
00167     row_(grad.row_),
00168     coeffs_interior_(grad.coeffs_interior_),
00169     prem_apps_(grad.prem_apps_),
00170     weights_crs_(grad.weights_crs_),
00171     weights_cbs_(grad.weights_cbs_),
00172     mim_bndy_(grad.mim_bndy_),
00173     gradient_(grad.gradient_),
00174     sums_rows_mim_bndy_(grad.sums_rows_mim_bndy_),
00175     mimetic_threshold_(grad.mimetic_threshold_) {}
00176
00177 mtk::Grad1D::~~Grad1D() {
00178
00179     delete[] coeffs_interior_;
00180     coeffs_interior_ = nullptr;
00181
00182     delete[] prem_apps_;
00183     prem_apps_ = nullptr;
00184
00185     delete[] weights_crs_;
00186     weights_crs_ = nullptr;
00187
00188     delete[] weights_cbs_;
00189     weights_cbs_ = nullptr;
00190
00191     delete[] mim_bndy_;
00192     mim_bndy_ = nullptr;
00193
00194     delete[] gradient_;
00195     gradient_ = nullptr;
00196 }
00197
00198 bool mtk::Grad1D::ConstructGrad1D(int order_accuracy,
00199     Real mimetic_threshold) {
00200
00201     #ifdef MTK_PERFORM_PREVENTIONS
00202     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00203     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00204     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00205         __FILE__, __LINE__, __func__);
00206
00207     if (order_accuracy >= mtk::kCriticalOrderAccuracyGrad) {
00208         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00209     }
00210
00211     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00212     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00213     #endif
00214
00215     order_accuracy_ = order_accuracy;
00216     mimetic_threshold_ = mimetic_threshold;
00217
00218     bool abort_construction = ComputeStencilInteriorGrid();
00219
00220     #ifdef MTK_PERFORM_PREVENTIONS
00221     if (!abort_construction) {
00222         std::cerr << "Could NOT complete stage 1." << std::endl;
00223         std::cerr << "Exiting..." << std::endl;
00224         return false;
00225     }
00226     #endif
00227
00228     // At this point, we already have the values for the interior stencil stored
00229     // in the coeffs_interior_ array.
00230
00231     dim_null_ = order_accuracy_/2 - 1;

```

```

00232
00233 num_bndy_approxs_ = dim_null_ + 1;
00234
00235 #ifdef MTK_PRECISION_DOUBLE
00236 num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00237 #else
00238 num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00239 #endif
00240
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 // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00254 //
00255 // However, given the nature of the Vandermonde matrices we've just
00256 // computed, they all possess the same null-space. Therefore, we impose the
00257 // convention of computing the null-space of the first Vandermonde matrix
00258 // (west boundary).
00259
00260 // In the case of the gradient, the first Vandermonde system has a unique
00261 // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00262 // matrix used to assemble said system, will have an empty null-space.
00263
00264 // Therefore, we only compute a rational basis for the case of order higher
00265 // than second.
00266
00267 if (dim_null_ > 0) {
00268
00269     abort_construction = ComputeRationalBasisNullSpace();
00270
00271     #ifdef MTK_PERFORM_PREVENTIONS
00272     if (!abort_construction) {
00273         std::cerr << "Could NOT complete stage 2.1." << std::endl;
00274         std::cerr << "Exiting..." << std::endl;
00275         return false;
00276     }
00277     #endif
00278 }
00279
00281 abort_construction = ComputePreliminaryApproximations();
00282
00283 #ifdef MTK_PERFORM_PREVENTIONS
00284 if (!abort_construction) {
00285     std::cerr << "Could NOT complete stage 2.2." << std::endl;
00286     std::cerr << "Exiting..." << std::endl;
00287     return false;
00288 }
00289 #endif
00290
00292 abort_construction = ComputeWeights();
00293
00294 #ifdef MTK_PERFORM_PREVENTIONS
00295 if (!abort_construction) {
00296     std::cerr << "Could NOT complete stage 2.3." << std::endl;
00297     std::cerr << "Exiting..." << std::endl;
00298     return false;
00299 }
00300 #endif
00301
00303 if (dim_null_ > 0) {
00304
00305     abort_construction = ComputeStencilBoundaryGrid();
00306
00307     #ifdef MTK_PERFORM_PREVENTIONS
00308     if (!abort_construction) {
00309         std::cerr << "Could NOT complete stage 2.4." << std::endl;
00310         std::cerr << "Exiting..." << std::endl;
00311         return false;
00312     }
00313     #endif
00314 }
00315
00317

```

```

00318 // Once we have the following three collections of data:
00319 // (a) the coefficients for the interior,
00320 // (b) the coefficients for the boundary (if it applies),
00321 // (c) and the weights (if it applies),
00322 // we will store everything in the output array:
00323
00324 abort_construction = AssembleOperator();
00325
00326 #ifdef MTK_PERFORM_PREVENTIONS
00327 if (!abort_construction) {
00328     std::cerr << "Could NOT complete stage 3." << std::endl;
00329     std::cerr << "Exiting..." << std::endl;
00330     return false;
00331 }
00332 #endif
00333
00334 return true;
00335 }
00336
00337 int mtk::Grad1D::num_bndy_coeffs() const {
00338
00339     return num_bndy_coeffs_;
00340 }
00341
00342 mtk::Real *mtk::Grad1D::coeffs_interior() const {
00343
00344     return coeffs_interior_;
00345 }
00346
00347 mtk::Real *mtk::Grad1D::weights_crs() const {
00348
00349     return weights_crs_;
00350 }
00351
00352 mtk::Real *mtk::Grad1D::weights_cbs() const {
00353
00354     return weights_cbs_;
00355 }
00356
00357 mtk::DenseMatrix mtk::Grad1D::mim_bndy() const {
00358
00359     mtk::DenseMatrix xx(dim_null_ + 1, 3*order_accuracy_/2);
00360
00361     auto counter = 0;
00362     for (auto ii = 0; ii < dim_null_ + 1; ++ii) {
00363         for (auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00364             xx.SetValue(ii, jj, gradient_[2*order_accuracy_ + 1 + counter]);
00365             counter++;
00366         }
00367     }
00368
00369     return xx;
00370 }
00371
00372 std::vector<mtk::Real> mtk::Grad1D::sums_rows_mim_bndy() const {
00373
00374     return sums_rows_mim_bndy_;
00375 }
00376
00377 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00378     mtk::Real west,
00379     mtk::Real east,
00380     int num_cells_x) const {
00381
00382     int nn{num_cells_x}; // Number of cells on the grid.
00383
00384     #ifdef MTK_PERFORM_PREVENTIONS
00385     mtk::Tools::Prevent(east < west, __FILE__, __LINE__, __func__);
00386     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00387     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00388     #endif
00389
00390     mtk::Real delta_x = (east - west)/((mtk::Real) num_cells_x);
00391
00392     mtk::Real inv_delta_x{mtk::kOne/delta_x};
00393
00394     int gg_num_rows = nn + 1;
00395     int gg_num_cols = nn + 2;
00396     int elements_per_row = num_bndy_coeffs_;
00397     int num_extra_rows = order_accuracy_/2;

```

```

00398 // Output matrix featuring sizes for gradient operators.
00399 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00400
00402
00403 auto ee_index = 0;
00404 for (auto ii = 0; ii < num_extra_rows; ii++) {
00405     auto cc = 0;
00406     for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00407         if(cc >= elements_per_row) {
00408             out.SetValue(ii, jj, mtk::kZero);
00409         } else {
00410             out.SetValue(ii, jj,
00411                 gradient_[2*order_accuracy_ + 1 + ee_index++] * inv_delta_x);
00412             cc++;
00413         }
00414     }
00415 }
00416
00418
00419 for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00420     auto jj = ii - num_extra_rows + 1;
00421     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00422         out.SetValue(ii, jj, coeffs_interior_[cc] * inv_delta_x);
00423     }
00424 }
00425
00427
00428 ee_index = 0;
00429 for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00430     auto cc = 0;
00431     for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00432         if(cc >= elements_per_row) {
00433             out.SetValue(ii, jj, mtk::kZero);
00434         } else {
00435             out.SetValue(ii, jj,
00436                 -gradient_[2*order_accuracy_ + 1 +
00437 ee_index++] * inv_delta_x);
00438             cc++;
00439         }
00440     }
00441 }
00442
00443 return out;
00444 }
00445
00446 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00447     const UniStgGrid1D &grid) const {
00448
00449     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00450
00451 #ifdef MTK_PERFORM_PREVENTIONS
00452     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00453     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00454 #endif
00455
00456     mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00457
00458     int gg_num_rows = nn + 1;
00459     int gg_num_cols = nn + 2;
00460     int elements_per_row = num_bndy_coeffs_;
00461     int num_extra_rows = order_accuracy_/2;
00462
00463 // Output matrix featuring sizes for gradient operators.
00464 mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00465
00467
00468 auto ee_index = 0;
00469 for (auto ii = 0; ii < num_extra_rows; ii++) {
00470     auto cc = 0;
00471     for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00472         if(cc >= elements_per_row) {
00473             out.SetValue(ii, jj, mtk::kZero);
00474         } else {
00475             out.SetValue(ii, jj,
00476                 gradient_[2*order_accuracy_ + 1 + ee_index++] * inv_delta_x);
00477             cc++;
00478         }
00479     }
00480 }
00481
00483

```

```

00484     for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00485         auto jj = ii - num_extra_rows + 1;
00486         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00487             out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00488         }
00489     }
00490
00491     ee_index = 0;
00492     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00493         auto cc = 0;
00494         for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00495             if(cc >= elements_per_row) {
00496                 out.SetValue(ii, jj, mtk::kZero);
00497             } else {
00498                 out.SetValue(ii, jj,
00499                     -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00500                 cc++;
00501             }
00502         }
00503     }
00504     return out;
00505 }
00506
00507 mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix
00508 (
00509     int num_cells_x) const {
00510     int nn{num_cells_x}; // Number of cells on the grid.
00511
00512     #ifdef MTK_PERFORM_PREVENTIONS
00513     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00514     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00515     #endif
00516
00517     int gg_num_rows = nn + 1;
00518     int gg_num_cols = nn + 2;
00519     int elements_per_row = num_bndy_coeffs_;
00520     int num_extra_rows = order_accuracy_/2;
00521
00522     // Output matrix featuring sizes for gradient operators.
00523     mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00524
00525     auto ee_index = 0;
00526     for (auto ii = 0; ii < num_extra_rows; ii++) {
00527         auto cc = 0;
00528         for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00529             if(cc >= elements_per_row) {
00530                 out.SetValue(ii, jj, mtk::kZero);
00531             } else {
00532                 out.SetValue(ii, jj,
00533                     gradient_[2*order_accuracy_ + 1 + ee_index++]);
00534                 cc++;
00535             }
00536         }
00537     }
00538
00539     for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00540         auto jj = ii - num_extra_rows + 1;
00541         for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00542             out.SetValue(ii, jj, coeffs_interior_[cc]);
00543         }
00544     }
00545
00546     ee_index = 0;
00547     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00548         auto cc = 0;
00549         for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00550             if(cc >= elements_per_row) {
00551                 out.SetValue(ii, jj, mtk::kZero);
00552             } else {
00553                 out.SetValue(ii, jj,
00554                     -gradient_[2*order_accuracy_ + 1 + ee_index++]);
00555                 cc++;
00556             }
00557         }
00558     }
00559 }

```



```

00568
00569     return out;
00570 }
00571
00572 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00573
00574     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00575
00576     try {
00577         pp = new mtk::Real[order_accuracy_];
00578     } catch (std::bad_alloc &memory_allocation_exception) {
00579         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00580             std::endl;
00581         std::cerr << memory_allocation_exception.what() << std::endl;
00582     }
00583     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00584
00585     #ifdef MTK_PRECISION_DOUBLE
00586     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00587     #else
00588     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00589     #endif
00590
00591     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00592         pp[ii] = pp[ii - 1] + mtk::kOne;
00593     }
00594
00595     #if MTK_VERBOSE_LEVEL > 3
00596     std::cout << "pp =" << std::endl;
00597     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00598         std::cout << std::setw(12) << pp[ii];
00599     }
00600     std::cout << std::endl << std::endl;
00601     #endif
00602
00603     bool transpose{false};
00604
00605     mtk::DenseMatrix vander_matrix(pp, order_accuracy_, order_accuracy_, transpose);
00606
00607     #if MTK_VERBOSE_LEVEL > 4
00608     std::cout << "vander_matrix = " << std::endl;
00609     std::cout << vander_matrix << std::endl << std::endl;
00610     #endif
00611
00612     try {
00613         coeffs_interior_ = new mtk::Real[order_accuracy_];
00614     } catch (std::bad_alloc &memory_allocation_exception) {
00615         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00616             std::endl;
00617         std::cerr << memory_allocation_exception.what() << std::endl;
00618     }
00619     memset(coeffs_interior_, mtk::kZero,
00620         sizeof(coeffs_interior_[0])*order_accuracy_);
00621
00622     coeffs_interior_[1] = mtk::kOne;
00623
00624     #if MTK_VERBOSE_LEVEL > 3
00625     std::cout << "oo =" << std::endl;
00626     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00627         std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00628     }
00629     std::cout << std::endl;
00630     #endif
00631
00632     int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00633         coeffs_interior_)};
00634
00635     #ifdef MTK_PERFORM_PREVENTIONS
00636     if (!info) {
00637         std::cout << "System solved! Interior stencil attained!" << std::endl;
00638         std::cout << std::endl;
00639     }
00640     else {
00641         std::cerr << "Something wrong solving system! info = " << info << std::endl;
00642         std::cerr << "Exiting..." << std::endl;
00643         return false;
00644     }
00645 }

```

```

00653     #endif
00654
00655     #if MTK_VERBOSE_LEVEL > 3
00656     std::cout << "coeffs_interior_" << std::endl;
00657     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00658         std::cout << std::setw(12) << coeffs_interior_[ii];
00659     }
00660     std::cout << std::endl << std::endl;
00661     #endif
00662
00663     delete [] pp;
00664     pp = nullptr;
00665
00666     return true;
00667 }
00668
00669 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00670
00671     mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00672
00673     try {
00674         gg = new mtk::Real[num_bndy_coeffs_];
00675     } catch (std::bad_alloc &memory_allocation_exception) {
00676         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00677             std::endl;
00678         std::cerr << memory_allocation_exception.what() << std::endl;
00679     }
00680     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00681
00682     #ifdef MTK_PRECISION_DOUBLE
00683     gg[1] = 1.0/2.0;
00684     #else
00685     gg[1] = 1.0f/2.0f;
00686     #endif
00687     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00688         gg[ii] = gg[ii - 1] + mtk::kOne;
00689     }
00690
00691     #if MTK_VERBOSE_LEVEL > 3
00692     std::cout << "gg =" << std::endl;
00693     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00694         std::cout << std::setw(12) << gg[ii];
00695     }
00696     std::cout << std::endl << std::endl;
00697     #endif
00698
00699     bool tran{true}; // Should I transpose the Vandermonde matrix.
00700
00701     mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00702
00703     #if MTK_VERBOSE_LEVEL > 4
00704     std::cout << "aa_west_t =" << std::endl;
00705     std::cout << aa_west_t << std::endl;
00706     #endif
00707
00708     mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00709         (aa_west_t));
00710
00711     #if MTK_VERBOSE_LEVEL > 3
00712     std::cout << "qq_t =" << std::endl;
00713     std::cout << qq_t << std::endl;
00714     #endif
00715
00716     int kk_num_rows{num_bndy_coeffs_};
00717     int kk_num_cols{dim_null_};
00718
00719     mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00720
00721     // In the case of the gradient, even though we must solve for a null-space
00722     // of dimension 2, we must only extract ONE basis for the kernel.
00723     // We perform this extraction here:
00724
00725     int aux_{kk_num_rows - kk_num_cols};
00726     for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00727         aux_--;
00728         for (auto jj = 0; jj < kk_num_rows; jj++) {
00729             kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =

```

```

00737         qq_t.data()[ii*num_bndy_coeffs_ + jj];
00738     }
00739 }
00740
00741 #if MTK_VERBOSE_LEVEL > 2
00742 std::cout << "kk =" << std::endl;
00743 std::cout << kk << std::endl;
00744 std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00745 std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00746 std::cout << std::endl;
00747 #endif
00748
00750
00751 // Scale thus requesting that the last entries of the attained basis for the
00752 // null-space, adopt the pattern we require.
00753 // Essentially we will implement the following MATLAB pseudo-code:
00754 // scalers = kk(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00755 // SK = kk*scalers
00756 // where SK is the scaled null-space.
00757
00758 // In this point, we almost have all the data we need correctly allocated
00759 // in memory. We will create the matrix iden_, and elements we wish to scale
00760 // in the kk array. Using the concept of the leading dimension, we could just
00761 // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00762 // GET how does it work. So I will just create a matrix with the content of
00763 // this array that we need, solve for the scalers and then scale the
00764 // whole kk:
00765
00766 // We will then create memory for that sub-matrix of kk (subk).
00767
00768 mtk::DenseMatrix subk(dim_null_, dim_null_);
00769
00770 auto zz = 0;
00771 for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00772     for (auto jj = 0; jj < dim_null_; jj++) {
00773         subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00774     }
00775     zz++;
00776 }
00777
00778 #if MTK_VERBOSE_LEVEL > 4
00779 std::cout << "subk =" << std::endl;
00780 std::cout << subk << std::endl;
00781 #endif
00782
00783 subk.Transpose();
00784
00785 #if MTK_VERBOSE_LEVEL > 4
00786 std::cout << "subk_t =" << std::endl;
00787 std::cout << subk << std::endl;
00788 #endif
00789
00790 bool padded{false};
00791 tran = false;
00792
00793 mtk::DenseMatrix iden(dim_null_, padded, tran);
00794
00795 #if MTK_VERBOSE_LEVEL > 4
00796 std::cout << "iden =" << std::endl;
00797 std::cout << iden << std::endl;
00798 #endif
00799
00800 // Solve the system to compute the scalers.
00801 // An example of the system to solve, for k = 8, is:
00802 //
00803 // subk*scalers = iden or
00804 //
00805 // | 0.386018 -0.0339244 -0.129478 |           | 1 0 0 |
00806 // | -0.119774 0.0199423 0.0558632 |*scalers = | 0 1 0 |
00807 // | 0.0155708 -0.00349546 -0.00853182 |       | 0 0 1 |
00808 //
00809 // Notice this is a nrhs = 3 system.
00810 // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00811 // will be stored in the created identity matrix.
00812 // Let us first transpose subk (because of LAPACK):
00813
00814 int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00815
00816 #ifdef MTK_PERFORM_PREVENTIONS
00817 if (!info) {
00818     std::cout << "System successfully solved!" <<

```

```

00819     std::endl;
00820 } else {
00821     std::cerr << "Something went wrong solving system! info = " << info <<
00822     std::endl;
00823     std::cerr << "Exiting..." << std::endl;
00824     return false;
00825 }
00826 std::cout << std::endl;
00827 #endif
00828
00829 #if MTK_VERBOSE_LEVEL > 4
00830 std::cout << "Computed scalars:" << std::endl;
00831 std::cout << iden << std::endl;
00832 #endif
00833
00834 // Multiply the two matrices to attain a scaled basis for null-space.
00835
00836 rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00837
00838 #if MTK_VERBOSE_LEVEL > 4
00839 std::cout << "Rational basis for the null-space:" << std::endl;
00840 std::cout << rat_basis_null_space_ << std::endl;
00841 #endif
00842
00843 // At this point, we have a rational basis for the null-space, with the
00844 // pattern we need! :)
00845
00846 delete [] gg;
00847 gg = nullptr;
00848
00849 return true;
00850 }
00851
00852 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00853
00854     mtk::Real *gg{}; // Generator vector for the first approximation.
00855
00856     try {
00857         gg = new mtk::Real[num_bndy_coeffs_];
00858     } catch (std::bad_alloc &memory_allocation_exception) {
00859         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00860         std::endl;
00861         std::cerr << memory_allocation_exception.what() << std::endl;
00862     }
00863     memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00864
00865     #ifdef MTK_PRECISION_DOUBLE
00866     gg[1] = 1.0/2.0;
00867     #else
00868     gg[1] = 1.0f/2.0f;
00869     #endif
00870     for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00871         gg[ii] = gg[ii - 1] + mtk::kOne;
00872     }
00873
00874     #if MTK_VERBOSE_LEVEL > 3
00875     std::cout << "gg0 =" << std::endl;
00876     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00877         std::cout << std::setw(12) << gg[ii];
00878     }
00879     std::cout << std::endl << std::endl;
00880     #endif
00881
00882     // Allocate 2D array to store the collection of preliminary approximations.
00883     try {
00884         prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00885     } catch (std::bad_alloc &memory_allocation_exception) {
00886         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00887         std::endl;
00888         std::cerr << memory_allocation_exception.what() << std::endl;
00889     }
00890     memset(prem_apps_,
00891            mtk::kZero,
00892            sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00893
00894     for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00895
00896         // Re-check new generator vector for every iteration except for the first.
00897         #if MTK_VERBOSE_LEVEL > 3

```

```

00902     if (ll > 0) {
00903         std::cout << "gg_" << ll << " =" << std::endl;
00904         for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00905             std::cout << std::setw(12) << gg[ii];
00906         }
00907         std::cout << std::endl << std::endl;
00908     }
00909 #endif
00910
00911 bool transpose{false};
00912
00913 mtk::DenseMatrix aa(gg,
00914                     num_bndy_coeffs_, order_accuracy_ + 1,
00915                     transpose);
00916
00917 #if MTK_VERBOSE_LEVEL > 4
00918 std::cout << "aa_" << ll << " =" << std::endl;
00919 std::cout << aa << std::endl;
00920 #endif
00921
00922 mtk::Real *ob{};
00923
00924 auto ob_ld = num_bndy_coeffs_;
00925
00926 try {
00927     ob = new mtk::Real[ob_ld];
00928 } catch (std::bad_alloc &memory_allocation_exception) {
00929     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00930         std::endl;
00931     std::cerr << memory_allocation_exception.what() << std::endl;
00932 }
00933 memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00934
00935 ob[1] = mtk::kOne;
00936
00937 #if MTK_VERBOSE_LEVEL > 3
00938 std::cout << "ob = " << std::endl << std::endl;
00939 for (auto ii = 0; ii < ob_ld; ++ii) {
00940     std::cout << std::setw(12) << ob[ii] << std::endl;
00941 }
00942 std::cout << std::endl;
00943 #endif
00944
00945 // However, this is an under-determined system of equations. So we can not
00946 // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00947 // our LAPACKAdapter class.
00948
00949 int info_{
00950     mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
00951 , ob_ld)};
00952
00953 #ifdef MTK_PERFORM_PREVENTIONS
00954 if (!info_) {
00955     std::cout << "System successfully solved!" << std::endl << std::endl;
00956 } else {
00957     std::cerr << "Error solving system! info = " << info_ << std::endl;
00958     return false;
00959 }
00960 #endif
00961
00962 #if MTK_VERBOSE_LEVEL > 3
00963 std::cout << "ob =" << std::endl;
00964 for (auto ii = 0; ii < ob_ld; ++ii) {
00965     std::cout << std::setw(12) << ob[ii] << std::endl;
00966 }
00967 std::cout << std::endl;
00968 #endif
00969
00970 // This implies a DAXPY operation. However, we must construct the arguments
00971 // for this operation.
00972
00973 // Save them into the ob_bottom array:
00974
00975 Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00976
00977 try {
00978     ob_bottom = new mtk::Real[dim_null_];
00979 }

```

```

00987     } catch (std::bad_alloc &memory_allocation_exception) {
00988         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00989             std::endl;
00990         std::cerr << memory_allocation_exception.what() << std::endl;
00991     }
00992     memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00993
00994     for (auto ii = 0; ii < dim_null_; ++ii) {
00995         ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00996     }
00997
00998     #if MTK_VERBOSE_LEVEL > 3
00999     std::cout << "ob_bottom =" << std::endl;
01000     for (auto ii = 0; ii < dim_null_; ++ii) {
01001         std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
01002     }
01003     std::cout << std::endl;
01004     #endif
01005
01006     // We must computed an scaled ob, sob, using the scaled null-space in
01007     // rat_basis_null_space_.
01008     // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
01009     // or:                      ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
01010     // thus:                      Y =      a*A      *x      +      b*Y (DAXPY).
01011
01012     #if MTK_VERBOSE_LEVEL > 4
01013     std::cout << "Rational basis for the null-space:" << std::endl;
01014     std::cout << rat_basis_null_space_ << std::endl;
01015     #endif
01016
01017     mtk::Real alpha{-mtk::kOne};
01018     mtk::Real beta{mtk::kOne};
01019
01020     mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
01021                                     ob_bottom, beta, ob);
01022
01023     #if MTK_VERBOSE_LEVEL > 3
01024     std::cout << "scaled ob:" << std::endl;
01025     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01026         std::cout << std::setw(12) << ob[ii] << std::endl;
01027     }
01028     std::cout << std::endl;
01029     #endif
01030
01031     // We save the recently scaled solution, into an array containing these.
01032     // We can NOT start building the pi matrix, simply because I want that part
01033     // to be separated since its construction depends on the algorithm we want
01034     // to implement.
01035
01036     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01037         prem_apps_[ii*num_bndy_approxs_ + 11] = ob[ii];
01038     }
01039
01040     // After the first iteration, simply shift the entries of the last
01041     // generator vector used:
01042     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01043         gg[ii]--;
01044     }
01045
01046     // Garbage collection for this loop:
01047     delete[] ob;
01048     ob = nullptr;
01049
01050     delete[] ob_bottom;
01051     ob_bottom = nullptr;
01052 } // End of: for (ll = 0; ll < dim_null; ll++);
01053
01054 #if MTK_VERBOSE_LEVEL > 4
01055 std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
01056 for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01057     for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01058         std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
01059     }
01060     std::cout << std::endl;
01061 }
01062 std::cout << std::endl;
01063 #endif
01064
01065 delete[] gg;
01066 gg = nullptr;

```

```

01069
01070     return true;
01071 }
01072
01073 bool mtk::Grad1D::ComputeWeights() {
01074
01075     // Matrix to compute the weights as in the CRSA.
01076     mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
01077
01078     // Assemble the pi matrix using:
01079     // 1. The collection of scaled preliminary approximations.
01080     // 2. The collection of coefficients approximating at the interior.
01081     // 3. The scaled basis for the null-space.
01082
01083     // 1.1. Process array of scaled preliminary approximations.
01084
01085     // These are queued in scaled_solutions. Each one of these, will be a column
01086     // of the pi matrix:
01087     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01088         for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01089             pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01090                 prem_apps_[ii*num_bndy_approxs_ + jj];
01091         }
01092     }
01093
01094     // 1.2. Add columns from known stencil approximating at the interior.
01095
01096     // However, these must be padded by zeros, according to their position in the
01097     // final pi matrix:
01098     auto mm = 1;
01099     for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
01100         for (auto ii = 0; ii < order_accuracy_; ++ii) {
01101             auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
01102                 (order_accuracy_/2 + 1)) + jj;
01103             pi.data()[de] = coeffs_interior_[ii];
01104         }
01105         ++mm;
01106     }
01107
01108     rat_basis_null_space_.OrderColMajor();
01109
01110     #if MTK_VERBOSE_LEVEL > 4
01111     std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01112     std::cout << rat_basis_null_space_ << std::endl;
01113     #endif
01114
01115     // 1.3. Add final set of columns: rational basis for null-space.
01116
01117     for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01118         jj < num_bndy_coeffs_ - 1; ++jj) {
01119         for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01120             auto og =
01121                 (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01122             auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01123             pi.data()[de] = rat_basis_null_space_.data()[og];
01124         }
01125     }
01126
01127     #if MTK_VERBOSE_LEVEL > 4
01128     std::cout << "coeffs_interior_" << std::endl;
01129     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01130         std::cout << std::setw(12) << coeffs_interior_[ii];
01131     }
01132     std::cout << std::endl << std::endl;
01133     #endif
01134
01135     #if MTK_VERBOSE_LEVEL > 4
01136     std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01137     std::cout << pi << std::endl;
01138     #endif
01139
01140     // This imposes the mimetic condition.
01141
01142     mtk::Real *hh{}; // Right-hand side to compute weights in the C{R,B}SA.
01143
01144     try {
01145         hh = new mtk::Real[num_bndy_coeffs_];
01146     } catch (std::bad_alloc &memory_allocation_exception) {
01147         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<

```

```

01152         std::endl;
01153         std::cerr << memory_allocation_exception.what() << std::endl;
01154     }
01155     memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01156
01157     hh[0] = -mtk::kOne;
01158     for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01159         auto aux_xx = mtk::kZero;
01160         for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01161             aux_xx += coeffs_interior_[jj];
01162         }
01163         hh[ii] = -mtk::kOne*aux_xx;
01164     }
01165
01166     // That is, we construct a system, to solve for the weights.
01167
01168     // Once again we face the challenge of solving with LAPACK. However, for the
01169     // CRSA, this matrix PI is over-determined, since it has more rows than
01170     // unknowns. However, according to the theory, the solution to this system is
01171     // unique. We will use dgels_.
01172
01173     try {
01174         weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01175     } catch (std::bad_alloc &memory_allocation_exception) {
01176         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01177             std::endl;
01178         std::cerr << memory_allocation_exception.what() << std::endl;
01179     }
01180     memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01181
01182     int weights_ld{pi.num_cols() + 1};
01183
01184     // Preserve hh.
01185     std::copy(hh, hh + weights_ld, weights_cbs_);
01186
01187     pi.Transpose();
01188
01189     int info{
01190         mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01191             weights_cbs_, weights_ld)
01192     };
01193
01194     #ifdef MTK_PERFORM_PREVENTIONS
01195     if (!info) {
01196         std::cout << "System successfully solved!" << std::endl << std::endl;
01197     } else {
01198         std::cerr << "Error solving system! info = " << info << std::endl;
01199         return false;
01200     }
01201     #endif
01202
01203     #if MTK_VERBOSE_LEVEL > 3
01204     std::cout << "hh =" << std::endl;
01205     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01206         std::cout << std::setw(11) << hh[ii] << std::endl;
01207     }
01208     std::cout << std::endl;
01209     #endif
01210
01211     // Preserve the original weights for research.
01212
01213     try {
01214         weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01215     } catch (std::bad_alloc &memory_allocation_exception) {
01216         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01217             std::endl;
01218         std::cerr << memory_allocation_exception.what() << std::endl;
01219     }
01220     memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01221
01222     std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01223
01224     #if MTK_VERBOSE_LEVEL > 3
01225     std::cout << "weights_CRSA + lambda =" << std::endl;
01226     for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01227         std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01228     }
01229     std::cout << std::endl;
01230     #endif
01231
01232
01233

```



```

01235
01236 if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01237
01238
01239     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01240
01241
01242     // 6.1. Insert preliminary approximations to first set of columns.
01243
01244     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01245         for (auto jj = 0; jj < num_bndy_approx_ + 1; ++jj) {
01246             phi.data()[ii*(order_accuracy_ + 1) + jj] =
01247                 prem_apps[ii*num_bndy_approx_ + jj];
01248         }
01249     }
01250
01251     // 6.2. Skip a column and negate preliminary approximations.
01252
01253     for (auto jj = 0; jj < order_accuracy_ + 1; jj++) {
01254         for (auto ii = 1; ii < num_bndy_approx_ + 1; ii++) {
01255             auto de = (ii + order_accuracy_ - num_bndy_approx_ + jj*order_accuracy_);
01256             auto og = (num_bndy_approx_ - ii + (jj)*num_bndy_approx_);
01257             phi.data()[de] = -pre_apps[og];
01258         }
01259     }
01260
01261     // 6.3. Flip negative columns up-down.
01262
01263     for (auto ii = 0; ii < order_accuracy_/2; ii++) {
01264         for (auto jj = num_bndy_approx_ + 1; jj < order_accuracy_; jj++) {
01265             auto aux = phi.data()[ii*order_accuracy_ + jj];
01266             phi.data()[ii*order_accuracy_ + jj] =
01267                 phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01268             phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01269         }
01270     }
01271
01272     // 6.4. Insert stencil.
01273
01274     auto mm = 0;
01275     for (auto jj = num_bndy_approx_; jj < num_bndy_approx_ + 1; jj++) {
01276         for (auto ii = 0; ii < order_accuracy_ + 1; ii++) {
01277             if (ii == 0) {
01278                 phi.data()[jj] = 0.0;
01279             } else {
01280                 phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior[ii - 1];
01281             }
01282         }
01283         mm++;
01284     }
01285
01286     #if MTK_VERBOSE_LEVEL > 4
01287     std::cout << "phi =" << std::endl;
01288     std::cout << phi << std::endl;
01289     #endif
01290
01291     mtk::Real *lamed{}; // Used to build big lambda.
01292
01293     try {
01294         lamed = new mtk::Real[num_bndy_approx_ - 1];
01295     } catch (std::bad_alloc &memory_allocation_exception) {
01296         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01297             std::endl;
01298         std::cerr << memory_allocation_exception.what() << std::endl;
01299     }
01300     memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approx_ - 1));
01301
01302     for (auto ii = 0; ii < num_bndy_approx_ - 1; ++ii) {
01303         lamed[ii] = hh[ii + order_accuracy_ + 1];
01304     }
01305
01306     #if MTK_VERBOSE_LEVEL > 3
01307     std::cout << "lamed =" << std::endl;
01308     for (auto ii = 0; ii < num_bndy_approx_ - 1; ++ii) {
01309         std::cout << std::setw(12) << lamed[ii] << std::endl;
01310     }
01311     std::cout << std::endl;
01312     #endif
01313
01314     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01315         mtk::Real temp = mtk::kZero;
01316     }
01317

```

```

01318         for(auto jj = 0; jj < num_bndy_approx_ - 1; ++jj) {
01319             temp = temp +
01320                 lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01321         }
01322         hh[ii] = hh[ii] - temp;
01323     }
01324
01325     #if MTK_VERBOSE_LEVEL > 3
01326     std::cout << "big_lambda =" << std::endl;
01327     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01328         std::cout << std::setw(12) << hh[ii] << std::endl;
01329     }
01330     std::cout << std::endl;
01331     #endif
01332
01333     #ifdef MTK_VERBOSE_WEIGHTS
01334     int copy_result{1};
01335     #else
01336     int copy_result{};
01337     #endif
01338
01339     int minrow_{std::numeric_limits<int>::infinity()};
01340
01341     mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01342         order_accuracy_)};
01343     mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01344
01345     mtk::Real normerr_; // Norm of the error for the solution on each row.
01346
01347     #ifdef MTK_VERBOSE_WEIGHTS
01348     std::ofstream table("grad_ld_" + std::to_string(order_accuracy_) +
01349         "_weights.tex");
01350
01351     table << "\\begin{tabular}{c}{c}";
01352     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01353         table << 'c';
01354     }
01355     table << "c\\n\\toprule\\nRow & ";
01356     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01357         table << "$ q_{" + std::to_string(ii) + "}$ & ";
01358     }
01359     table << " Relative error \\|\\|\\|\\n\\midrule\\n";
01360     #endif
01361
01362     for(auto row_ = 0; row_ < order_accuracy_ + 1; ++row_) {
01363         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
01364             data(),
01365                 order_accuracy_ + 1,
01366                 order_accuracy_,
01367                 order_accuracy_,
01368                 hh,
01369                 weights_cbs_,
01370                 row_,
01371                 mimetic_threshold_,
01372                 copy_result);
01373         mtk::Real aux{normerr_/norm};
01374
01375         #if MTK_VERBOSE_LEVEL > 2
01376         std::cout << "Relative norm: " << aux << " " << std::endl;
01377         std::cout << std::endl;
01378         #endif
01379
01380         if (aux < minnorm) {
01381             minnorm = aux;
01382             minrow_ = row_;
01383         }
01384
01385         #ifdef MTK_VERBOSE_WEIGHTS
01386         table << std::to_string(row_ + 1) << " & ";
01387         if (normerr_ != std::numeric_limits<mtk::Real>::infinity()) {
01388             for (int ii = 1; ii <= order_accuracy_; ++ii) {
01389                 table << std::to_string(weights_cbs_[ii - 1]) + " & ";
01390             }
01391             table << std::to_string(aux) << " \\|\\|\\| " << std::endl;
01392         } else {
01393             table << "\\multicolumn{" << std::to_string(order_accuracy_) <<
01394                 "}{}{\\emptyset$} & ";
01395             table << " - \\|\\|\\| " << std::endl;
01396         }
01397         #endif

```

```

01398     }
01399
01400     #ifdef MTK_VERBOSE_WEIGHTS
01401     table << "\\midrule" << std::endl;
01402     table << "CRS weights:";
01403     for (int ii = 1; ii <= order_accuracy_; ++ii) {
01404         table << " & " << std::to_string(weights_crs_[ii - 1]);
01405     }
01406     table << " & - \\|\\|\\|\\bottomrule\\n\\end{tabular}" << std::endl;
01407     table.close();
01408     #endif
01409
01410     #if MTK_VERBOSE_LEVEL > 3
01411     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01412         std::endl;
01413     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01414         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01415     }
01416     std::cout << std::endl;
01417     #endif
01418
01420
01421     // After we know which row yields the smallest relative norm that row is
01422     // chosen to be the objective function and the result of the optimizer is
01423     // chosen to be the new weights_.
01424
01425     #if MTK_VERBOSE_LEVEL > 2
01426     std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01427         minrow_ + 1 << std::endl;
01428     std::cout << std::endl;
01429     #endif
01430
01431     copy_result = 1;
01432     normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
data(),
01433                                     order_accuracy_ + 1,
01434                                     order_accuracy_,
01435                                     order_accuracy_,
01436                                     hh,
01437                                     weights_cbs_,
01438                                     minrow_,
01439                                     mimetic_threshold_,
01440                                     copy_result);
01441     mtk::Real aux_{normerr_/norm};
01442     #if MTK_VERBOSE_LEVEL > 2
01443     std::cout << "Relative norm: " << aux_ << std::endl;
01444     std::cout << std::endl;
01445     #endif
01446
01447     delete [] lamed;
01448     lamed = nullptr;
01449 }
01450
01451 delete [] hh;
01452 hh = nullptr;
01453
01454 return true;
01455 }
01456
01457 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01458
01459     #if MTK_VERBOSE_LEVEL > 3
01460     std::cout << "weights_* + lambda =" << std::endl;
01461     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01462         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01463     }
01464     std::cout << std::endl;
01465     #endif
01466
01468
01469     mtk::Real *lambda{}; // Collection of bottom values from weights_.
01470
01471     try {
01472         lambda = new mtk::Real[dim_null_];
01473     } catch (std::bad_alloc &memory_allocation_exception) {
01474         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01475             std::endl;
01476         std::cerr << memory_allocation_exception.what() << std::endl;
01477     }
01478     memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01479

```

```

01480     for (auto ii = 0; ii < dim_null_; ++ii) {
01481         lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01482     }
01483
01484     #if MTK_VERBOSE_LEVEL > 3
01485     std::cout << "lambda =" << std::endl;
01486     for (auto ii = 0; ii < dim_null_; ++ii) {
01487         std::cout << std::setw(12) << lambda[ii] << std::endl;
01488     }
01489     std::cout << std::endl;
01490     #endif
01491
01492
01493
01494     mtk::Real *alpha{}; // Collection of alpha values.
01495
01496     try {
01497         alpha = new mtk::Real[dim_null_];
01498     } catch (std::bad_alloc &memory_allocation_exception) {
01499         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01500             std::endl;
01501         std::cerr << memory_allocation_exception.what() << std::endl;
01502     }
01503     memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01504
01505     for (auto ii = 0; ii < dim_null_; ++ii) {
01506         alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01507     }
01508
01509     #if MTK_VERBOSE_LEVEL > 3
01510     std::cout << "alpha =" << std::endl;
01511     for (auto ii = 0; ii < dim_null_; ++ii) {
01512         std::cout << std::setw(12) << alpha[ii] << std::endl;
01513     }
01514     std::cout << std::endl;
01515     #endif
01516
01517
01518
01519     try {
01520         mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01521     } catch (std::bad_alloc &memory_allocation_exception) {
01522         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01523             std::endl;
01524         std::cerr << memory_allocation_exception.what() << std::endl;
01525     }
01526     memset(mim_bndy_,
01527         mtk::kZero,
01528         sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01529
01530     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01531         for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {
01532             mim_bndy_[ii*num_bndy_approxs_ + jj] =
01533                 prem_apps_[ii*num_bndy_approxs_ + jj] +
01534                 alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01535         }
01536     }
01537
01538     for(auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01539         mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01540             prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01541     }
01542
01543     #if MTK_VERBOSE_LEVEL > 4
01544     std::cout << "Collection of mimetic approximations:" << std::endl;
01545     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01546         for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01547             std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01548         }
01549         std::cout << std::endl;
01550     }
01551     std::cout << std::endl;
01552     #endif
01553
01554
01555
01556     for (auto ii = 0; ii < num_bndy_approxs_; ++ii) {
01557         sums_rows_mim_bndy_.push_back(mtk::kZero);
01558         for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01559             sums_rows_mim_bndy_[ii] += mim_bndy_[jj*num_bndy_approxs_ + ii];
01560         }
01561     }
01562
01563     #if MTK_VERBOSE_LEVEL > 3

```

```

01564     std::cout << "Row-wise sum of mimetic approximations:" << std::endl;
01565     for (auto ii = 0; ii < num_bndy_approxs_; ++ii) {
01566         std::cout << std::setw(13) << sums_rows_mim_bndy_[ii];
01567     }
01568     std::cout << std::endl;
01569     std::cout << std::endl;
01570     #endif
01571
01572     delete[] lambda;
01573     lambda = nullptr;
01574
01575     delete[] alpha;
01576     alpha = nullptr;
01577
01578     return true;
01579 }
01580
01581 bool mtk::Grad1D::AssembleOperator(void) {
01582
01583     // The output array will have this form:
01584     // 1. The first entry of the array will contain the used order kk.
01585     // 2. The second entry of the array will contain the collection of
01586     // approximating coefficients for the interior of the grid.
01587     // 3. The third entry will contain a collection of weights.
01588     // 4. The next dim_null - 1 entries will contain the collections of
01589     // approximating coefficients for the west boundary of the grid.
01590
01591     gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01592         num_bndy_approxs_*num_bndy_coeffs_;
01593
01594     #if MTK_VERBOSE_LEVEL > 2
01595     std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01596     #endif
01597
01598     try {
01599         gradient_ = new mtk::Real[gradient_length_];
01600     } catch (std::bad_alloc &memory_allocation_exception) {
01601         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01602             std::endl;
01603         std::cerr << memory_allocation_exception.what() << std::endl;
01604     }
01605     memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01606
01607
01608
01609     gradient_[0] = order_accuracy_;
01610
01611
01612
01613     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01614         gradient_[ii + 1] = coeffs_interior_[ii];
01615     }
01616
01617
01618
01619     for (auto ii = 0; ii < order_accuracy_; ++ii) {
01620         gradient_[order_accuracy_ + 1 + ii] = weights_cbs_[ii];
01621     }
01622
01623
01624
01625
01626     int offset{2*order_accuracy_ + 1};
01627
01628
01629     int aux {}; // Auxiliary variable.
01630
01631     if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01632         for (auto ii = 0; ii < num_bndy_approxs_; ++ii) {
01633             for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01634                 gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01635                 aux++;
01636             }
01637         }
01638     } else {
01639         gradient_[offset + 0] = prem_apps_[0];
01640         gradient_[offset + 1] = prem_apps_[1];
01641         gradient_[offset + 2] = prem_apps_[2];
01642     }
01643
01644     #if MTK_VERBOSE_LEVEL > 1
01645     std::cout << "1D " << order_accuracy_ << "-order grad built!" << std::endl;
01646     std::cout << std::endl;
01647     #endif
01648
01649     return true;
01650 }

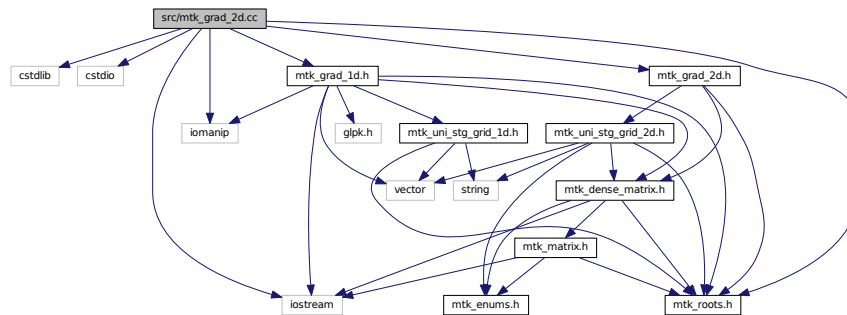
```

18.95 src/mtk_grad_2d.cc File Reference

Implements the class Grad2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_2d.h"
```

Include dependency graph for mtk_grad_2d.cc:



18.95.1 Detailed Description

This class implements a 2D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_2d.cc](#).

18.96 mtk_grad_2d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
```

```

00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_2d.h"
00066
00067 mtk::Grad2D::Grad2D():
00068     order_accuracy_(),
00069     mimetic_threshold_() {}
00070
00071 mtk::Grad2D::Grad2D(const Grad2D &grad):
00072     order_accuracy_(grad.order_accuracy_),
00073     mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad2D::~~Grad2D() {}
00076
00077 bool mtk::Grad2D::ConstructGrad2D(const
    mtk::UniStgGrid2D &grid,
                                int order_accuracy,
                                mtk::Real mimetic_threshold) {
00078
00079
00080
00081     int num_cells_x = grid.num_cells_x();
00082     int num_cells_y = grid.num_cells_y();
00083
00084     int mx = num_cells_x + 1; // Gx vertical dimension
00085     int nx = num_cells_x + 2; // Gx horizontal dimension
00086     int my = num_cells_y + 1; // Gy vertical dimension
00087     int ny = num_cells_y + 2; // Gy horizontal dimension
00088
00089     mtk::Grad1D grad;
00090
00091     bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00092
00093     #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 }

```

18.97 src/mtk_grad_3d.cc File Reference

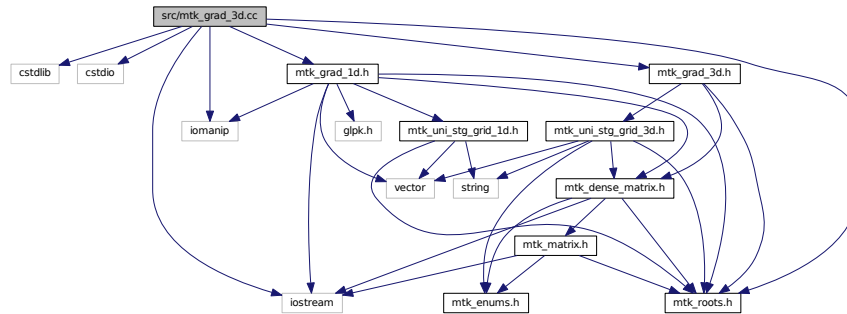
Implements the class Grad3D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_3d.h"

```


Include dependency graph for mtk_grad_3d.cc:



18.97.1 Detailed Description

This class implements a 3D gradient operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (C↔BSA).

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_3d.cc](#).

18.98 mtk_grad_3d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.

```

```

00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_3d.h"
00066
00067 mtk::Grad3D::Grad3D():
00068     order_accuracy_(),
00069     mimetic_threshold_() {}
00070
00071 mtk::Grad3D::Grad3D(const Grad3D &grad):
00072     order_accuracy_(grad.order_accuracy_),
00073     mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad3D::~Grad3D() {}
00076
00077 bool mtk::Grad3D::ConstructGrad3D(const
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.east_bndy();
00107     auto bottom = grid.bottom_bndy();
00108     auto top = grid.top_bndy();
00109
00110     mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00111     mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00112     mtk::UniStgGrid1D grid_z(bottom, top, num_cells_z);
00113
00114     mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00115     mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00116     mtk::DenseMatrix Gz(grad.ReturnAsDenseMatrix(grid_z));
00117
00118     bool padded{true};
00119     bool transpose{true};
00120
00121     mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00122     mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00123     mtk::DenseMatrix tiz(num_cells_z, padded, transpose);

```

```

00124
00126
00127 mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(tiz, tiy));
00128 mtk::DenseMatrix gx(mtk::DenseMatrix::Kron(aux1, Gx));
00129
00131
00132 mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(tiz, Gy));
00133 mtk::DenseMatrix gy(mtk::DenseMatrix::Kron(aux2, tix));
00134
00136
00137 mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Gz, tiy));
00138 mtk::DenseMatrix gz(mtk::DenseMatrix::Kron(aux3, tix));
00139
00140 #if MTK_VERBOSE_LEVEL > 2
00141 std::cout << "Gx: " << mx << " by " << nx << std::endl;
00142 std::cout << "Transpose Ix: " << num_cells_x << " by " << nx << std::endl;
00143 std::cout << "Gy: " << my << " by " << ny << std::endl;
00144 std::cout << "Transpose Iy: " << num_cells_y << " by " << ny << std::endl;
00145 std::cout << "Gz: " << mz << " by " << nz << std::endl;
00146 std::cout << "Transpose Iz: " << num_cells_z << " by " << nz << std::endl;
00147 #endif
00148
00150
00151 int total_rows{mx*num_cells_y*num_cells_z +
00152               num_cells_x*my*num_cells_z +
00153               num_cells_x*num_cells_y*mz};
00154 int total_cols{nx*ny*nz};
00155
00156 #if MTK_VERBOSE_LEVEL > 2
00157 std::cout << "Grad 3D: " << total_rows << " by " << total_cols << std::endl;
00158 #endif
00159
00160 mtk::DenseMatrix g3d(total_rows, total_cols);
00161
00162 for(auto ii = 0; ii < total_cols; ii++) {
00163     for(auto jj = 0; jj < mx*num_cells_y*num_cells_z; jj++) {
00164         g3d.SetValue(jj, ii, gx.GetValue(jj, ii));
00165     }
00166
00167     int offset = mx*num_cells_y*num_cells_z;
00168
00169     for(auto kk = 0; kk < num_cells_x*my*num_cells_z; kk++) {
00170         g3d.SetValue(kk + offset, ii, gy.GetValue(kk, ii));
00171     }
00172
00173     offset += num_cells_x*my*num_cells_z;
00174
00175     for(auto ll = 0; ll < num_cells_x*num_cells_y*mz; ll++) {
00176         g3d.SetValue(ll + offset, ii, gz.GetValue(ll, ii));
00177     }
00178 }
00179
00180 gradient_ = g3d;
00181
00182 return info;
00183 }
00184
00185 mtk::DenseMatrix mtk::Grad3D::ReturnAsDenseMatrix() const {
00186
00187     return gradient_;
00188 }

```

18.99 src/mtk_interp_1d.cc File Reference

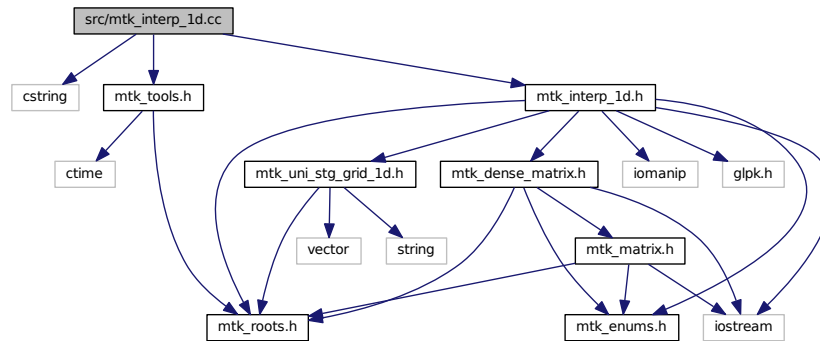
Includes the implementation of the class Interp1D.

```

#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)`

18.99.1 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](#).

18.100 mtk_interp_1d.cc

```

00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
  
```

```

00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #include <cstring>
00059
00060 #include "mtk_tools.h"
00061
00062 #include "mtk_interp_1d.h"
00063
00064 namespace mtk {
00065
00066 std::ostream& operator <<(std::ostream &stream, mtk::Interp1D &in) {
00067
00068
00069     stream << "coeffs_interior_[1:" << in.order_accuracy_ << "] = ";
00070     for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00071         stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00072     }
00073     stream << std::endl;
00074
00075     return stream;
00076 }
00077
00078 }
00079
00080 mtk::Interp1D::Interp1D():
00081     dir_interp_(mtk::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
00107
00108 }

```

```

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
00134  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
00162
00163     out.SetValue(0, 0, mtk::kOne);
00164
00166
00167     for (auto ii = 1; ii < gg_num_rows - 1; ++ii) {
00168         for (auto jj = ii; jj < order_accuracy_ + ii; ++jj) {
00169             out.SetValue(ii, jj, mtk::kOne/order_accuracy_);
00170         }
00171     }
00172
00174
00175     out.SetValue(gg_num_rows - 1, gg_num_cols - 1, mtk::kOne);
00176
00177     return out;
00178 }

```

18.101 src/mtk_lap_1d.cc File Reference

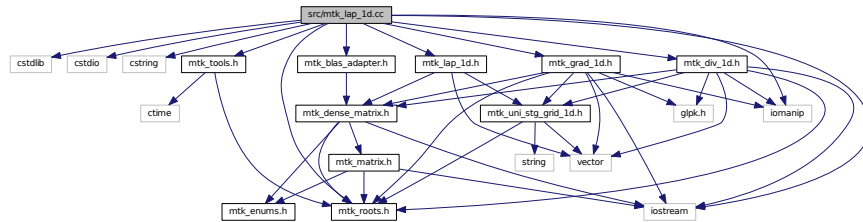
Includes the implementation of the class Lap1D.

```

#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"

```

Include dependency graph for mtk_lap_1d.cc:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::Lap1D &in)`

18.101.1 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](#).

18.102 mtk_lap_1d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,

```

```

00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059 #include <cstring>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_tools.h"
00066 #include "mtk_blas_adapter.h"
00067 #include "mtk_grad_ld.h"
00068 #include "mtk_div_ld.h"
00069 #include "mtk_lap_ld.h"
00070
00071 namespace mtk {
00072
00073 std::ostream& operator <<(std::ostream &stream, mtk::Lap1D &in) {
00074
00075     int output_precision{4};
00076     int output_width{8};
00077
00078     stream << "Order of accuracy: " << in.laplacian_[0] << std::endl;
00079
00080     stream << "Interior stencil: " << std::endl;
00081     for (auto ii = 1; ii <= (2*in.order_accuracy_ - 1); ++ii) {
00082         stream << std::setprecision(output_precision) << std::setw(output_width) <<
00083             in.laplacian_[ii] << ' ';
00084     }
00085     stream << std::endl;
00086
00087     auto offset = 1 + (2*in.order_accuracy_ - 1);
00088     for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00089         stream << "Mimetic boundary row " << ii + 1 << ":" << std::endl;
00090         for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00091             stream << std::setprecision(output_precision) <<
00092                 std::setw(output_width) <<

```



```

00100         in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj] << ' ';
00101     }
00102     stream << std::endl;
00103     stream << "Sum of elements in row " << ii + 1 << ": " <<
00104         in.sums_rows_mim_bndy_[ii];
00105     stream << std::endl;
00106 }
00107
00108 return stream;
00109 }
00110 }
00111
00112 mtk::Lap1D::Lap1D():
00113     order_accuracy_(mtk::kDefaultOrderAccuracy),
00114     laplacian_length_(),
00115     delta_(mtk::kZero),
00116     mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00117
00118 mtk::Lap1D::~~Lap1D() {
00119
00120     delete [] laplacian_;
00121     laplacian_ = nullptr;
00122 }
00123
00124 int mtk::Lap1D::order_accuracy() const {
00125
00126     return order_accuracy_;
00127 }
00128
00129 mtk::Real mtk::Lap1D::mimetic_threshold() const {
00130
00131     return mimetic_threshold_;
00132 }
00133
00134 mtk::Real mtk::Lap1D::delta() const {
00135
00136     return delta_;
00137 }
00138
00139 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00140                                 mtk::Real mimetic_threshold) {
00141
00142     #ifdef MTK_PERFORM_PREVENTIONS
00143     mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00144     mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00145     mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00146                         __FILE__, __LINE__, __func__);
00147
00148     if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00149         std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00150     }
00151
00152     std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00153     std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00154     #endif
00155
00156     order_accuracy_ = order_accuracy;
00157     mimetic_threshold_ = mimetic_threshold;
00158
00159     mtk::Grad1D grad; // Mimetic gradient.
00160
00161     bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00162
00163     #ifdef MTK_PERFORM_PREVENTIONS
00164     if (!info) {
00165         std::cerr << "Mimetic grad could not be built." << std::endl;
00166         return false;
00167     }
00168     #endif
00169
00170     mtk::Div1D div; // Mimetic divergence.
00171
00172     info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00173
00174     #ifdef MTK_PERFORM_PREVENTIONS
00175     if (!info) {
00176         std::cerr << "Mimetic div could not be built." << std::endl;
00177         return false;
00178     }
00179     #endif
00180
00181     return info;
00182 }

```

```

00183
00184
00185 // Since these are mimetic operator, we must multiply the matrices arising
00186 // from both the divergence and the Laplacian, in order to get the
00187 // approximating coefficients for the Laplacian operator.
00188
00189 // However, we must choose a grid that implied a step size of 1, so to get
00190 // the approximating coefficients, without being affected from the
00191 // normalization with respect to the grid (dimensionless).
00192
00193 // Also, the grid must be of the minimum size to support the requested order
00194 // of accuracy. We must please the divergence for this!
00195
00196 mtk::UniStgGrid1D aux(mtk::kZero,
00197                      (mtk::Real) 3*order_accuracy_ - 1,
00198                      3*order_accuracy_ - 1);
00199
00200 #if MTK_VERBOSE_LEVEL > 2
00201 std::cout << "aux =" << std::endl;
00202 std::cout << aux << std::endl;
00203 std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00204 std::cout << std::endl;
00205 #endif
00206
00207 mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00208
00209 #if MTK_VERBOSE_LEVEL > 4
00210 std::cout << "grad_m =" << std::endl;
00211 std::cout << grad_m << std::endl;
00212 #endif
00213
00214 mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux));
00215
00216 #if MTK_VERBOSE_LEVEL > 4
00217 std::cout << "div_m =" << std::endl;
00218 std::cout << div_m << std::endl;
00219 #endif
00220
00221 mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00222
00223 lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00224
00225 #if MTK_VERBOSE_LEVEL > 4
00226 std::cout << "lap =" << std::endl;
00227 std::cout << lap << std::endl;
00228 #endif
00229
00230 // The output array will have this form:
00231 // 1. The first entry of the array will contain the used order kk.
00232 // 2. The second entry of the array will contain the collection of
00233 // approximating coefficients for the interior of the grid.
00234 // 3. The next entries will contain the collections of approximating
00235 // coefficients for the west boundary of the grid.
00236
00237 laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00238                   (order_accuracy_ - 1)*(2*order_accuracy_);
00239
00240 #if MTK_VERBOSE_LEVEL > 2
00241 std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00242 std::cout << std::endl;
00243 #endif
00244
00245 try {
00246     laplacian_ = new mtk::Real[laplacian_length_];
00247 } catch (std::bad_alloc &memory_allocation_exception) {
00248     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00249     std::endl;
00250     std::cerr << memory_allocation_exception.what() << std::endl;
00251 }
00252
00253 memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00254
00255 laplacian_[0] = order_accuracy_;
00256
00257 for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00258     laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00259 }
00260

```

```

00273
00275
00276   auto offset = 1 + (2*order_accuracy_ - 1);
00277
00278   for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00279       sums_rows_mim_bndy_.push_back(mtk::kZero);
00280       for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00281           register mtk::Real aux{lap.GetValue(1 + ii, jj)};
00282           laplacian_[offset + ii*(2*order_accuracy_ + jj)] = aux;
00283           sums_rows_mim_bndy_[ii] += aux;
00284       }
00285   }
00286
00287   delta_ = mtk::kZero;
00288
00289   return true;
00290 }
00291
00292 std::vector<mtk::Real> mtk::Lap1D::sums_rows_mim_bndy() const {
00293
00294     return sums_rows_mim_bndy_;
00295 }
00296
00297 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(
00298     const UniStgGrid1D &grid) const {
00299
00300     int nn{grid.num_cells_x()}; // Number of cells on the grid.
00301
00302     #ifdef MTK_PERFORM_PREVENTIONS
00303     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00304     mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00305     #endif
00306
00307     mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00308
00309     delta_ = grid.delta_x();
00310
00311     mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
00312     dx^2.
00313
00314
00315     auto offset = (1 + 2*order_accuracy_ - 1);
00316
00317     for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00318         for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00319             lap.SetValue(1 + ii,
00320                 jj,
00321                 idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00322         }
00323     }
00324
00325
00326
00327     offset = 1 + (order_accuracy_ - 1);
00328
00329     int kk{1};
00330     for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00331         int mm{1};
00332         for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00333             lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00334             mm = mm + 1;
00335         }
00336         kk = kk + 1;
00337     }
00338
00339
00340
00341     offset = (1 + 2*order_accuracy_ - 1);
00342
00343     auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00344
00345     auto ll = 1;
00346     auto rr = 1;
00347     for (auto ii = nn; ii > aux - 1; --ii) {
00348         auto cc = 0;
00349         for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00350             lap.SetValue(ii, jj, lap.GetValue(rr, cc));
00351             ++ll;
00352             ++cc;
00353         }
00354         rr++;
00355     }
00356

```

```

00363
00364     return lap;
00365 }
00366
00367 const mtk::Real* mtk::Lap1D::data(const UniStgGrid1D &grid) const {
00368
00369     mtk::DenseMatrix tmp;
00370
00371     tmp = ReturnAsDenseMatrix(grid);
00372
00373     return tmp.data();
00374 }

```

18.103 src/mtk_lap_2d.cc File Reference

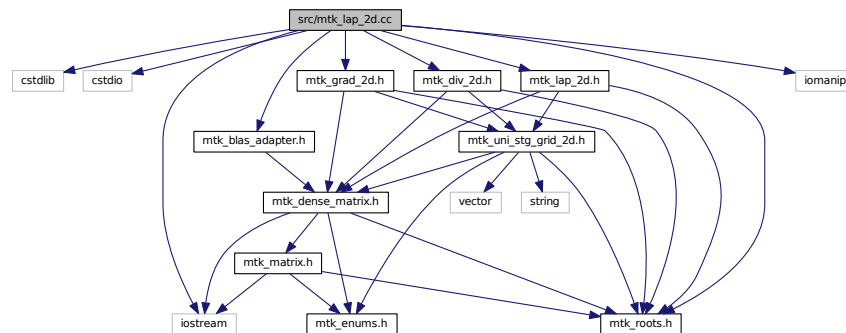
Includes the implementation of the class Lap2D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_lap_2d.h"

```

Include dependency graph for mtk_lap_2d.cc:



18.103.1 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](#).

18.104 mtk_lap_2d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_2d.h"
00066 #include "mtk_div_2d.h"
00067 #include "mtk_lap_2d.h"
00068
00069 mtk::Lap2D::Lap2D(): order_accuracy_(), mimetic_threshold_() {}
00070
00071 mtk::Lap2D::Lap2D(const Lap2D &lap):
00072     order_accuracy_(lap.order_accuracy_),
00073     mimetic_threshold_(lap.mimetic_threshold_) {}
00074
00075 mtk::Lap2D::~~Lap2D() {}
00076
00077 bool mtk::Lap2D::ConstructLap2D(const
00078     mtk::UniStgGrid2D &grid,
00079                             int order_accuracy,
00080                             mtk::Real mimetic_threshold) {
00081     mtk::Grad2D gg;
00082     mtk::Div2D dd;
00083
00084     bool info{gg.ConstructGrad2D(grid, order_accuracy, mimetic_threshold)};
00085
00086     #ifdef MTK_PERFORM_PREVENTIONS

```

```

00087     if (!info) {
00088         std::cerr << "Mimetic lap could not be built." << std::endl;
00089         return info;
00090     }
00091     #endif
00092
00093     info = dd.ConstructDiv2D(grid, order_accuracy, mimetic_threshold);
00094
00095     #ifdef MTK_PERFORM_PREVENTIONS
00096     if (!info) {
00097         std::cerr << "Mimetic div could not be built." << std::endl;
00098         return info;
00099     }
00100     #endif
00101
00102     mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00103     mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00104
00105     laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
00106
00107     return info;
00108 }
00109
00110 mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix() const {
00111     return laplacian_;
00112 }
00113
00114
00115 mtk::Real *mtk::Lap2D::data() const {
00116     return laplacian_.data();
00117 }
00118 }

```

18.105 src/mtk_lap_3d.cc File Reference

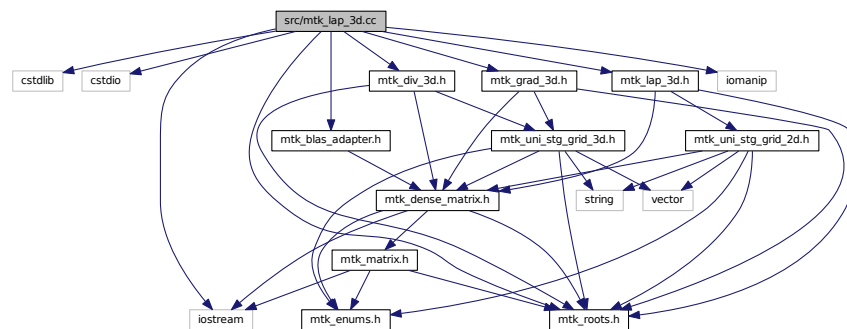
Includes the implementation of the class Lap3D.

```

#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_3d.h"
#include "mtk_div_3d.h"
#include "mtk_lap_3d.h"

```

Include dependency graph for mtk_lap_3d.cc:



18.105.1 Detailed Description

This class implements a 3D Laplacian operator, constructed using the Castillo-Blomgren-Sanchez (CBS) Algorithm (CBSA).

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lap_3d.cc](#).

18.106 mtk_lap_3d.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_3d.h"
00066 #include "mtk_div_3d.h"
00067 #include "mtk_lap_3d.h"
00068
00069 mtk::UniStgGrid3D mtk::Lap3D::operator*(const

```

```

    mtk::UniStgGrid3D &grid) const {
00070
00071     mtk::UniStgGrid3D out;
00072
00073     return out;
00074 }
00075
00076 mtk::Lap3D::Lap3D(): order_accuracy_(), mimetic_threshold_() {}
00077
00078 mtk::Lap3D::Lap3D(const Lap3D &lap):
00079     order_accuracy_(lap.order_accuracy_),
00080     mimetic_threshold_(lap.mimetic_threshold_) {}
00081
00082 mtk::Lap3D::~Lap3D() {}
00083
00084 bool mtk::Lap3D::ConstructLap3D(const
    mtk::UniStgGrid3D &grid,
00085                                int order_accuracy,
00086                                mtk::Real mimetic_threshold) {
00087
00088     mtk::Grad3D gg;
00089     mtk::Div3D dd;
00090
00091     bool info{gg.ConstructGrad3D(grid, order_accuracy, mimetic_threshold)};
00092
00093     #ifdef MTK_PERFORM_PREVENTIONS
00094     if (!info) {
00095         std::cerr << "Mimetic lap could not be built." << std::endl;
00096         return info;
00097     }
00098     #endif
00099
00100     info = dd.ConstructDiv3D(grid, order_accuracy, mimetic_threshold);
00101
00102     #ifdef MTK_PERFORM_PREVENTIONS
00103     if (!info) {
00104         std::cerr << "Mimetic div could not be built." << std::endl;
00105         return info;
00106     }
00107     #endif
00108
00109     mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00110     mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00111
00112     laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
00113
00114     return info;
00115 }
00116
00117 mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix() const {
00118
00119     return laplacian_;
00120 }
00121
00122 mtk::Real *mtk::Lap3D::data() const {
00123
00124     return laplacian_.data();
00125 }

```

18.107 src/mtk_lapack_adapter.cc File Reference

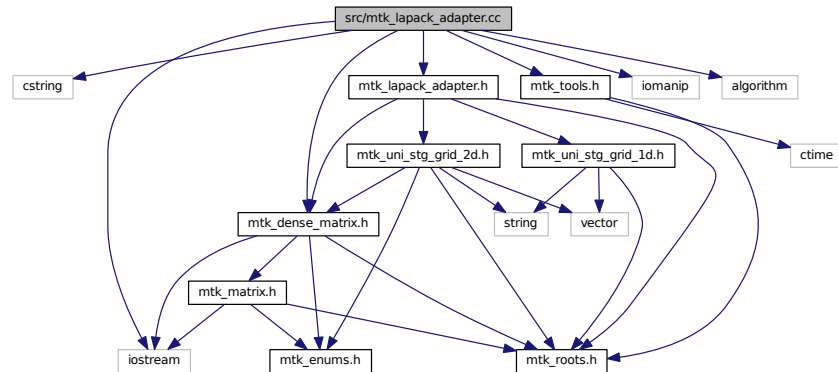
Adapter class for the LAPACK API.

```

#include <cstring>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_dense_matrix.h"
#include "mtk_lapack_adapter.h"

```


Include dependency graph for mtk_lapack_adapter.cc:



Namespaces

- [mtk](#)

Mimetic Methods Toolkit namespace.

Functions

- void [mtk::sgesv_](#) (int *n, int *nrhs, Real *a, int *lda, int *ipiv, Real *b, int *ldb, int *info)
- void [mtk::sgels_](#) (char *trans, int *m, int *n, int *nrhs, Real *a, int *lda, Real *b, int *ldb, Real *work, int *lwork, int *info)
Single-precision GEneral matrix Least Squares solver.
- void [mtk::sgeqrf_](#) (int *m, int *n, Real *a, int *lda, Real *tau, Real *work, int *lwork, int *info)
Single-precision GEneral matrix QR Factorization.
- void [mtk::sormqr_](#) (char *side, char *trans, int *m, int *n, int *k, Real *a, int *lda, Real *tau, Real *c, int *ldc, Real *work, int *lwork, int *info)
Single-precision Orthogonal [Matrix](#) from QR factorization.

18.107.1 Detailed Description

Implementation of a class that contains a collection of static member functions, that posses direct access to the underlying structure of the matrices, thus allowing programmers to exploit some of the numerical methods implemented in the LAPACK.

The **LAPACK (Linear Algebra PACKage)** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See also

<http://www.netlib.org/lapack/>

Todo Write documentation using LaTeX.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lapack_adapter.cc](#).

18.108 mtk_lapack_adapter.cc

```

00001
00022 /*
00023 Copyright (C) 2015, Computational Science Research Center, San Diego State
00024 University. All rights reserved.
00025
00026 Redistribution and use in source and binary forms, with or without modification,
00027 are permitted provided that the following conditions are met:
00028
00029 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00030 and a copy of the modified files should be reported once modifications are
00031 completed, unless these modifications are made through the project's GitHub
00032 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00033 should be developed and included in any deliverable.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions in binary form must reproduce the above copyright notice,
00039 this list of conditions and the following disclaimer in the documentation and/or
00040 other materials provided with the distribution.
00041
00042 4. Usage of the binary form on proprietary applications shall require explicit
00043 prior written permission from the the copyright holders, and due credit should
00044 be given to the copyright holders.
00045
00046 5. Neither the name of the copyright holder nor the names of its contributors
00047 may be used to endorse or promote products derived from this software without
00048 specific prior written permission.
00049
00050 The copyright holders provide no reassurances that the source code provided does
00051 not infringe any patent, copyright, or any other intellectual property rights of
00052 third parties. The copyright holders disclaim any liability to any recipient for
00053 claims brought against recipient by any third party for infringement of that
00054 parties intellectual property rights.
00055
00056 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00057 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00058 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00059 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00060 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00061 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00062 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00063 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00064 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00065 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00066 */
00067
00068 #include <cstring>
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <algorithm>
00074
00075 #include "mtk_tools.h"
00076 #include "mtk_dense_matrix.h"
00077 #include "mtk_lapack_adapter.h"
00078
00079 namespace mtk {
00080
00081 extern "C" {
00082
00083 #ifdef MTK_PRECISION_DOUBLE
00084
00103 void dgesv_(int* n,
00104             int* nrhs,
00105             Real* a,
00106             int* lda,

```

```
00107         int* ipiv,
00108         Real* b,
00109         int* ldb,
00110         int* info);
00111 #else
00112
00131 void sgesv_(int* n,
00132             int* nrhs,
00133             Real* a,
00134             int* lda,
00135             int* ipiv,
00136             Real* b,
00137             int* ldb,
00138             int* info);
00139 #endif
00140
00141 #ifdef MTK_PRECISION_DOUBLE
00142
00185 void dgels_(char* trans,
00186             int* m,
00187             int* n,
00188             int* nrhs,
00189             Real* a,
00190             int* lda,
00191             Real* b,
00192             int* ldb,
00193             Real* work,
00194             int* lwork,
00195             int* info);
00196 #else
00197
00240 void sgels_(char* trans,
00241             int* m,
00242             int* n,
00243             int* nrhs,
00244             Real* a,
00245             int* lda,
00246             Real* b,
00247             int* ldb,
00248             Real* work,
00249             int* lwork,
00250             int* info);
00251 #endif
00252
00253 #ifdef MTK_PRECISION_DOUBLE
00254
00283 void dgeqrf_(int *m,
00284              int *n,
00285              Real *a,
00286              int *lda,
00287              Real *tau,
00288              Real *work,
00289              int *lwork,
00290              int *info);
00291 #else
00292
00321 void sgeqrf_(int *m,
00322              int *n,
00323              Real *a,
00324              int *lda,
00325              Real *tau,
00326              Real *work,
00327              int *lwork,
00328              int *info);
00329 #endif
00330
00331 #ifdef MTK_PRECISION_DOUBLE
00332
00366 void dormqr_(char *side,
00367              char *trans,
00368              int *m,
00369              int *n,
00370              int *k,
00371              Real *a,
00372              int *lda,
00373              Real *tau,
00374              Real *c,
00375              int *ldc,
00376              Real *work,
00377              int *lwork,
00378              int *info);
```

```

00379 #else
00380
00414 void sormqr_(char *side,
00415             char *trans,
00416             int *m,
00417             int *n,
00418             int *k,
00419             Real *a,
00420             int *lda,
00421             Real *tau,
00422             Real *c,
00423             int *ldc,
00424             Real *work,
00425             int *lwork,
00426             int *info);
00427 #endif
00428 }
00429 }
00430
00431 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
    mtk::Real *rhs) {
00432
00433     #ifdef MTK_PERFORM_PREVENTIONS
00434     mtk::Tools::Prevent(rhs == nullptr, __FILE__, __LINE__, __func__);
00435     #endif
00436
00437     int *ipiv{};           // Array for pivoting information.
00438     int nrhs{1};           // Number of right-hand sides.
00439     int info{};            // Status of the solution.
00440     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00441
00442     try {
00443         ipiv = new int[mm_rank];
00444     } catch (std::bad_alloc &memory_allocation_exception) {
00445         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00446             std::endl;
00447         std::cerr << memory_allocation_exception.what() << std::endl;
00448     }
00449     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00450
00451     int ldbb = mm_rank;
00452     int mm_ld = mm_rank;
00453
00454     #ifdef MTK_PRECISION_DOUBLE
00455     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00456     #else
00457     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00458     #endif
00459
00460     delete [] ipiv;
00461
00462     return info;
00463 }
00464
00465 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
    mtk::DenseMatrix &bb) {
00466
00467     int nrhs{bb.num_rows()}; // Number of right-hand sides.
00468
00469     #ifdef MTK_PERFORM_PREVENTIONS
00470     mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00471     #endif
00472
00473     int *ipiv{};           // Array for pivoting information.
00474     int info{};            // Status of the solution.
00475     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00476
00477     try {
00478         ipiv = new int[mm_rank];
00479     } catch (std::bad_alloc &memory_allocation_exception) {
00480         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00481             std::endl;
00482         std::cerr << memory_allocation_exception.what() << std::endl;
00483     }
00484     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00485
00486     int ldbb = mm_rank;
00487     int mm_ld = mm_rank;
00488
00489
00490

```

```

00491  #ifdef MTK_PRECISION_DOUBLE
00492  dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00493  #else
00494  fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &lddb, &info);
00495  #endif
00496
00497  delete [] ipiv;
00498
00499  // After output, the data in the matrix will be column-major ordered.
00500
00501  bb.SetOrdering(mtk::MatrixOrdering::COL_MAJOR);
00502
00503  #if MTK_VERBOSE_LEVEL > 12
00504  std::cout << "bb_col_maj_ord =" << std::endl;
00505  std::cout << bb << std::endl;
00506  #endif
00507
00508  bb.OrderRowMajor();
00509
00510  #if MTK_VERBOSE_LEVEL > 12
00511  std::cout << "bb_row_maj_ord =" << std::endl;
00512  std::cout << bb << std::endl;
00513  #endif
00514
00515  return info;
00516 }
00517
00518 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
00519                                     mtk::UniStgGrid1D &rhs) {
00520
00521     int nrhs{1}; // Number of right-hand sides.
00522
00523     int *ipiv{}; // Array for pivoting information.
00524     int info{}; // Status of the solution.
00525     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00526
00527     try {
00528         ipiv = new int[mm_rank];
00529     } catch (std::bad_alloc &memory_allocation_exception) {
00530         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00531             std::endl;
00532         std::cerr << memory_allocation_exception.what() << std::endl;
00533     }
00534     memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00535
00536     int lddb = mm_rank;
00537     int mm_ld = mm_rank;
00538
00539     mm.OrderColMajor();
00540
00541     #ifdef MTK_PRECISION_DOUBLE
00542     dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00543         rhs.discrete_field(), &lddb, &info);
00544     #else
00545     fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00546         rhs.discrete_field(), &lddb, &info);
00547     #endif
00548
00549     mm.OrderRowMajor();
00550
00551     delete [] ipiv;
00552
00553     return info;
00554 }
00555
00556 int mtk::LAPACKAdapter::SolveDenseSystem(
    mtk::DenseMatrix &mm,
00557                                     mtk::UniStgGrid2D &rhs) {
00558
00559     int nrhs{1}; // Number of right-hand sides.
00560
00561     int *ipiv{}; // Array for pivoting information.
00562     int info{}; // Status of the solution.
00563     int mm_rank{mm.num_rows()}; // Rank of the matrix.
00564
00565     try {
00566         ipiv = new int[mm_rank];
00567     } catch (std::bad_alloc &memory_allocation_exception) {
00568         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00569             std::endl;

```

```

00570     std::cerr << memory_allocation_exception.what() << std::endl;
00571 }
00572 memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00573
00574 int ldbb = mm_rank;
00575 int mm_ld = mm_rank;
00576
00577 mm.OrderColMajor();
00578
00579 #ifdef MTK_PRECISION_DOUBLE
00580 dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00581       rhs.discrete_field(), &ldbb, &info);
00582 #else
00583 fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00584       rhs.discrete_field(), &ldbb, &info);
00585 #endif
00586
00587 mm.OrderRowMajor();
00588
00589 delete [] ipiv;
00590
00591 return info;
00592 }
00593
00594 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
(mtk::DenseMatrix &aa) {
00595
00596     mtk::Real *work{}; // Working array.
00597     mtk::Real *tau{}; // Array for the Householder scalars.
00598
00599     // Prepare to factorize: allocate and inquire for the value of lwork.
00600     try {
00601         work = new mtk::Real[1];
00602     } catch (std::bad_alloc &memory_allocation_exception) {
00603         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00604             std::endl;
00605         std::cerr << memory_allocation_exception.what() << std::endl;
00606     }
00607     memset(work, mtk::kZero, sizeof(aa.data()[0])*1);
00608
00609     int lwork{-1};
00610     int info{};
00611
00612     int aa_num_cols = aa.num_cols();
00613     int aaT_num_rows = aa.num_cols();
00614     int aaT_num_cols = aa.num_rows();
00615
00616     #if MTK_VERBOSE_LEVEL > 12
00617     std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00618     std::cout << aa << std::endl;
00619     #endif
00620
00621     #ifdef MTK_PRECISION_DOUBLE
00622     dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00623         tau,
00624         work, &lwork, &info);
00625     #else
00626     fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00627         tau,
00628         work, &lwork, &info);
00629     #endif
00630
00631     if (info == 0) {
00632         lwork = (int) work[0];
00633     } else {
00634         std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00635             std::endl;
00636         std::cerr << "Exiting..." << std::endl;
00637     }
00638
00639     #if MTK_VERBOSE_LEVEL > 10
00640     std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00641         << std::endl;
00642     #endif
00643
00644     delete [] work;
00645     work = nullptr;
00646
00647     // Once we know lwork, we can actually invoke the factorization:
00648     try {
00649         work = new mtk::Real [lwork];

```

```

00650 } catch (std::bad_alloc &memory_allocation_exception) {
00651     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00652     std::endl;
00653     std::cerr << memory_allocation_exception.what() << std::endl;
00654 }
00655 memset(work, mtk::kZero, sizeof(work[0])*lwork);
00656
00657 int ltau = std::min(aaT_num_rows, aaT_num_cols);
00658
00659 try {
00660     tau = new mtk::Real [ltau];
00661 } catch (std::bad_alloc &memory_allocation_exception) {
00662     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00663     std::endl;
00664     std::cerr << memory_allocation_exception.what() << std::endl;
00665 }
00666 memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00667
00668 #ifdef MTK_PRECISION_DOUBLE
00669 dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00670     tau, work, &lwork, &info);
00671 #else
00672 fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00673     tau, work, &lwork, &info);
00674 #endif
00675
00676 #ifdef MTK_PERFORM_PREVENTIONS
00677 if (!info) {
00678     std::cout << "QR factorization completed!" << std::endl << std::endl;
00679 } else {
00680     std::cerr << "Error solving system! info = " << info << std::endl;
00681     std::cerr << "Exiting..." << std::endl;
00682 }
00683 #endif
00684
00685 #if MTK_VERBOSE_LEVEL > 12
00686 std::cout << "Input matrix AFTER QR factorization:" << std::endl;
00687 std::cout << aa << std::endl;
00688 #endif
00689
00690 // We now generate the real matrix Q with orthonormal columns. This has to
00691 // be done separately since the actual output of dgeqrf_ (AA_) represents
00692 // the orthogonal matrix Q as a product of min(aa_num_rows, aa_num_cols)
00693 // elementary Householder reflectors. Notice that we must re-inquire the new
00694 // value for lwork that is used.
00695
00696 bool padded{false};
00697
00698 bool transpose{false};
00699
00700 mtk::DenseMatrix QQ_(aa.num_cols(), padded, transpose);
00701
00702 #if MTK_VERBOSE_LEVEL > 12
00703 std::cout << "Initialized QQ_T: " << std::endl;
00704 std::cout << QQ_ << std::endl;
00705 #endif
00706
00707 // Assemble the QQ_ matrix:
00708 lwork = -1;
00709
00710 delete[] work;
00711 work = nullptr;
00712
00713 try {
00714     work = new mtk::Real[l];
00715 } catch (std::bad_alloc &memory_allocation_exception) {
00716     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00717     std::endl;
00718     std::cerr << memory_allocation_exception.what() <<
00719     std::endl;
00720 }
00721 memset(work, mtk::kZero, sizeof(work[0])*l);
00722
00723 char side_{'L'};
00724 char trans_{'N'};
00725
00726 int aux = QQ_.num_rows();
00727
00728 #ifdef MTK_PRECISION_DOUBLE
00729 dormqr_(&side_, &trans_,
00730     &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,

```

```

00731         QQ_.data(), &aux, work, &lwork, &info);
00732     #else
00733     formqr_(&side_, &trans_,
00734             &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00735             QQ_.data(), &aux, work, &lwork, &info);
00736     #endif
00737
00738     if (info == 0) {
00739         lwork = (int) work[0];
00740     } else {
00741         std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00742         std::cerr << "Exiting..." << std::endl;
00743     }
00744
00745     #if MTK_VERBOSE_LEVEL > 10
00746     std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00747         std::endl << std::endl;
00748     #endif
00749
00750     delete[] work;
00751     work = nullptr;
00752
00753     try {
00754         work = new mtk::Real[lwork];
00755     } catch (std::bad_alloc &memory_allocation_exception) {
00756         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00757             std::endl;
00758         std::cerr << memory_allocation_exception.what() << std::endl;
00759     }
00760     memset(work, mtk::kZero, sizeof(work[0])*lwork);
00761
00762     #ifdef MTK_PRECISION_DOUBLE
00763     dormqr_(&side_, &trans_,
00764             &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00765             QQ_.data(), &aux, work, &lwork, &info);
00766     #else
00767     formqr_(&side_, &trans_,
00768             &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00769             QQ_.data(), &aux, work, &lwork, &info);
00770     #endif
00771
00772     #ifdef MTK_PERFORM_PREVENTIONS
00773     if (!info) {
00774         std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00775     } else {
00776         std::cerr << "Something went wrong solving system! info = " << info <<
00777             std::endl;
00778         std::cerr << "Exiting..." << std::endl;
00779     }
00780     #endif
00781
00782     delete[] work;
00783     work = nullptr;
00784
00785     delete[] tau;
00786     tau = nullptr;
00787
00788     return QQ_;
00789 }
00790
00791 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
    mtk::DenseMatrix &aa,
00792
00793     mtk::Real *ob_,
00794     int ob_ld_) {
00795     // We first invoke the solver to query for the value of lwork. For this,
00796     // we must at least allocate enough space to allow access to WORK(1), or
00797     // work[0]:
00798
00799     // If LWORK = -1, then a workspace query is assumed; the routine only
00800     // calculates the optimal size of the WORK array, returns this value as
00801     // the first entry of the WORK array, and no error message related to
00802     // LWORK is issued by XERBLA.
00803
00804     mtk::Real *work{}; // Work array.
00805
00806     try {
00807         work = new mtk::Real[1];
00808     } catch (std::bad_alloc &memory_allocation_exception) {
00809         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00810             std::endl;

```



```

00811     std::cerr << memory_allocation_exception.what() << std::endl;
00812 }
00813 memset(work, mtk::kZero, sizeof(work[0])*1);
00814
00815 char trans_{'N'};
00816 int nrhs_{1};
00817 int info{0};
00818 int lwork{-1};
00819
00820 int AA_num_rows_ = aa.num_cols();
00821 int AA_num_cols_ = aa.num_rows();
00822 int AA_ld_ = std::max(1, aa.num_cols());
00823
00824 #ifdef MTK_PRECISION_DOUBLE
00825 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00826       ob_, &ob_ld_,
00827       work, &lwork, &info);
00828 #else
00829 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00830       ob_, &ob_ld_,
00831       work, &lwork, &info);
00832 #endif
00833
00834 if (info == 0) {
00835     lwork = (int) work[0];
00836 } else {
00837     std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00838     std::endl;
00839     std::cerr << "Exiting..." << std::endl;
00840     return info;
00841 }
00842
00843 #if MTK_VERBOSE_LEVEL > 10
00844 std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00845     std::endl << std::endl;
00846 #endif
00847
00848 // We then use lwork's new value to create the work array:
00849 delete[] work;
00850 work = nullptr;
00851
00852 try {
00853     work = new mtk::Real[lwork];
00854 } catch (std::bad_alloc &memory_allocation_exception) {
00855     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00856     std::cerr << memory_allocation_exception.what() << std::endl;
00857 }
00858 memset(work, 0.0, sizeof(work[0])*lwork);
00859
00860 // We now invoke the solver again:
00861 #ifdef MTK_PRECISION_DOUBLE
00862 dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00863       ob_, &ob_ld_,
00864       work, &lwork, &info);
00865 #else
00866 sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00867       ob_, &ob_ld_,
00868       work, &lwork, &info);
00869 #endif
00870
00871 delete [] work;
00872 work = nullptr;
00873
00874 return info;
00875 }

```

18.109 src/mtk_matrix.cc File Reference

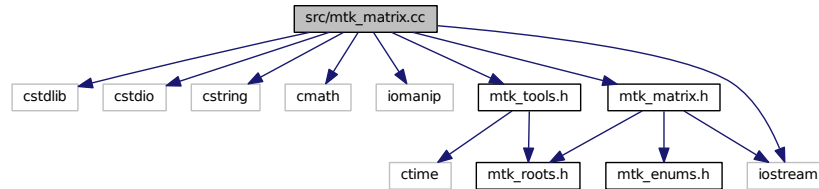
Implementing the representation of a matrix in the MTK.

```

#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <iostream>
#include "mtk_tools.h"
#include "mtk_matrix.h"

```

Include dependency graph for mtk_matrix.cc:



18.109.1 Detailed Description

Implementation of the representation for the matrices implemented in the MTK.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_matrix.cc](#).

18.110 mtk_matrix.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.

```

```

00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <cstdlib>
00057 #include <cstdio>
00058 #include <cstring>
00059 #include <cmath>
00060
00061 #include <iomanip>
00062 #include <iostream>
00063
00064 #include "mtk_tools.h"
00065 #include "mtk_matrix.h"
00066
00067 mtk::Matrix::Matrix():
00068     storage_(mtk::MatrixStorage::DENSE),
00069     ordering_(mtk::MatrixOrdering::ROW_MAJOR),
00070     num_rows_(),
00071     num_cols_(),
00072     num_values_(),
00073     ld_(),
00074     num_zero_(),
00075     num_non_zero_(),
00076     num_null_(),
00077     num_non_null_(),
00078     kl_(),
00079     ku_(),
00080     bandwidth_(),
00081     abs_density_(),
00082     rel_density_(),
00083     abs_sparsity_(),
00084     rel_sparsity_() {}
00085
00086 mtk::Matrix::Matrix(const Matrix &in):
00087     storage_(in.storage_),
00088     ordering_(in.ordering_),
00089     num_rows_(in.num_rows_),
00090     num_cols_(in.num_cols_),
00091     num_values_(in.num_values_),
00092     ld_(in.ld_),
00093     num_zero_(in.num_zero_),
00094     num_non_zero_(in.num_non_zero_),
00095     num_null_(in.num_null_),
00096     num_non_null_(in.num_non_null_),
00097     kl_(in.kl_),
00098     ku_(in.ku_),
00099     bandwidth_(in.bandwidth_),
00100     abs_density_(in.abs_density_),
00101     rel_density_(in.rel_density_),
00102     abs_sparsity_(in.abs_sparsity_),
00103     rel_sparsity_(in.rel_sparsity_) {}
00104
00105 mtk::Matrix::~Matrix() noexcept {}
00106
00107 mtk::MatrixStorage mtk::Matrix::storage() const noexcept {
00108
00109     return storage_;
00110 }
00111
00112 mtk::MatrixOrdering mtk::Matrix::ordering() const noexcept {
00113
00114     return ordering_;
00115 }
00116
00117 int mtk::Matrix::num_rows() const noexcept {

```

```

00118
00119     return num_rows_;
00120 }
00121
00122 int mtk::Matrix::num_cols() const noexcept {
00123
00124     return num_cols_;
00125 }
00126
00127 int mtk::Matrix::num_values() const noexcept {
00128
00129     return num_values_;
00130 }
00131
00132 int mtk::Matrix::ld() const noexcept {
00133
00134     return ld_;
00135 }
00136
00137 int mtk::Matrix::num_zero() const noexcept {
00138
00139     return num_zero_;
00140 }
00141
00142 int mtk::Matrix::num_non_zero() const noexcept {
00143
00144     return num_non_zero_;
00145 }
00146
00147 int mtk::Matrix::num_null() const noexcept {
00148
00149     return num_null_;
00150 }
00151
00152 int mtk::Matrix::num_non_null() const noexcept {
00153
00154     return num_non_null_;
00155 }
00156
00157 int mtk::Matrix::kl() const noexcept {
00158
00159     return kl_;
00160 }
00161
00162 int mtk::Matrix::ku() const noexcept {
00163
00164     return ku_;
00165 }
00166
00167 int mtk::Matrix::bandwidth() const noexcept {
00168
00169     return bandwidth_;
00170 }
00171
00172 mtk::Real mtk::Matrix::rel_density() const noexcept {
00173
00174     return rel_density_;
00175 }
00176
00177 mtk::Real mtk::Matrix::abs_sparsity() const noexcept {
00178
00179     return abs_sparsity_;
00180 }
00181
00182 mtk::Real mtk::Matrix::rel_sparsity() const noexcept {
00183
00184     return rel_sparsity_;
00185 }
00186
00187 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss)
noexcept {
00188
00189     #ifdef MTK_PERFORM_PREVENTIONS
00190     mtk::Tools::Prevent(!(ss == mtk::MatrixStorage::DENSE ||
00191                             ss == mtk::MatrixStorage::BANDED ||
00192                             ss == mtk::MatrixStorage::CRS),
00193                         __FILE__, __LINE__, __func__);
00194     #endif
00195
00196     storage_ = ss;
00197 }

```

```

00198
00199 void mtk::Matrix::set_ordering(const
    mtk::MatrixOrdering &oo) noexcept {
00200
00201     #ifdef MTK_PERFORM_PREVENTIONS
00202     bool aux{oo == mtk::MatrixOrdering::ROW_MAJOR ||
00203              oo == mtk::MatrixOrdering::COL_MAJOR};
00204     mtk::Tools::Prevent(!aux, __FILE__, __LINE__, __func__);
00205     #endif
00206
00207     ordering_ = oo;
00208
00209     ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00210           std::max(1,num_cols_): std::max(1,num_rows_);
00211 }
00212
00213 void mtk::Matrix::set_num_rows(const int &in) noexcept {
00214
00215     #ifdef MTK_PERFORM_PREVENTIONS
00216     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00217     #endif
00218
00219     num_rows_ = in;
00220     num_values_ = num_rows_*num_cols_;
00221     ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00222           std::max(1,num_cols_): std::max(1,num_rows_);
00223 }
00224
00225 void mtk::Matrix::set_num_cols(const int &in) noexcept {
00226
00227     #ifdef MTK_PERFORM_PREVENTIONS
00228     mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00229     #endif
00230
00231     num_cols_ = in;
00232     num_values_ = num_rows_*num_cols_;
00233     ld_ = (ordering_ == mtk::MatrixOrdering::ROW_MAJOR)?
00234           std::max(1,num_cols_): std::max(1,num_rows_);
00235 }
00236
00237 void mtk::Matrix::set_num_zero(const int &in) noexcept {
00238
00239     #ifdef MTK_PERFORM_PREVENTIONS
00240     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00241     #endif
00242
00243     num_zero_ = in;
00244     num_non_zero_ = num_values_ - num_zero_;
00245
00246     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00247     rel_sparsity_ = 1.0 - rel_density_;
00248 }
00249
00250
00251 void mtk::Matrix::set_num_null(const int &in) noexcept {
00252
00253     #ifdef MTK_PERFORM_PREVENTIONS
00254     mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00255     #endif
00256
00257     num_null_ = in;
00258     num_non_null_ = num_values_ - num_null_;
00259
00260     abs_density_ = (mtk::Real) num_non_null_/num_values_;
00261     abs_sparsity_ = 1.0 - abs_density_;
00262 }
00263
00264
00265 void mtk::Matrix::IncreaseNumZero() noexcept {
00266
00267     num_zero_++;
00268     num_non_zero_ = num_values_ - num_zero_;
00269     rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00270     rel_sparsity_ = 1.0 - rel_density_;
00271 }
00272
00273
00274
00275 void mtk::Matrix::IncreaseNumNull() noexcept {
00276
00277     num_null_++;
00278     num_non_null_ = num_values_ - num_null_;
00279     abs_density_ = (mtk::Real) num_non_null_/num_values_;
00280 }
00281

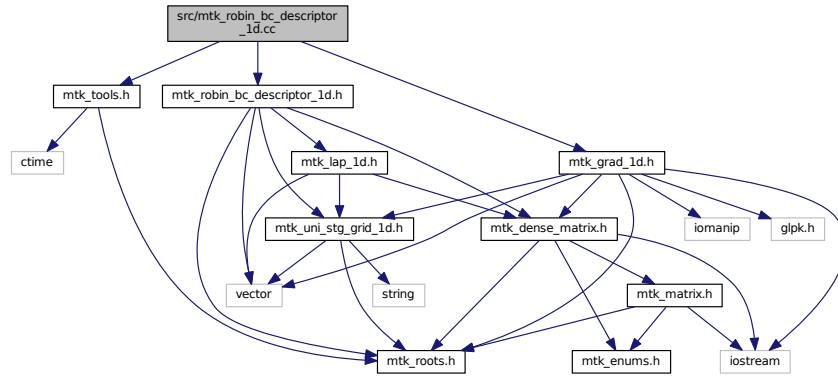
```

```
00282  abs_sparsity_ = 1.0 - abs_density_;
00283 }
```

18.111 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:
```



18.111.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 1D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

In a 1D context ($\partial\Omega = \{a, b\} \subset \mathbb{R}$), this condition can be written as follows:

$$\delta_a(a, t)u(a, t) - \eta_a(a, t)u'(a, t) = \beta_a(a, t),$$

$$\delta_b(b, t)u(b, t) + \eta_b(b, t)u'(b, t) = \beta_b(b, t).$$

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west and east, in 1D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the condition in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_robin_bc_descriptor_1d.cc](#).

18.112 mtk_robin_bc_descriptor_1d.cc

```

00001
00043 /*
00044 Copyright (C) 2015, Computational Science Research Center, San Diego State
00045 University. All rights reserved.
00046
00047 Redistribution and use in source and binary forms, with or without modification,
00048 are permitted provided that the following conditions are met:
00049
00050 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00051 and a copy of the modified files should be reported once modifications are
00052 completed, unless these modifications are made through the project's GitHub
00053 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00054 should be developed and included in any deliverable.
00055
00056 2. Redistributions of source code must be done through direct
00057 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00058
00059 3. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 4. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders, and due credit should
00065 be given to the copyright holders.
00066
00067 5. Neither the name of the copyright holder nor the names of its contributors
00068 may be used to endorse or promote products derived from this software without
00069 specific prior written permission.
00070
00071 The copyright holders provide no reassurances that the source code provided does
00072 not infringe any patent, copyright, or any other intellectual property rights of
00073 third parties. The copyright holders disclaim any liability to any recipient for
00074 claims brought against recipient by any third party for infringement of that
00075 parties intellectual property rights.
00076
00077 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00078 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00079 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00080 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00081 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00082 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00083 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00084 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00085 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00086 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00087 */
00088
00089 #include "mtk_tools.h"
00090 #include "mtk_grad_1d.h"
00091 #include "mtk_robin_bc_descriptor_1d.h"
00092
00093 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D():
00094     highest_order_diff_west_(-1),
00095     highest_order_diff_east_(-1),
00096     west_condition_(nullptr),
00097     east_condition_(nullptr) {}
00098
00099 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D(
00100     const mtk::RobinBCDescriptor1D &desc):
00101     highest_order_diff_west_(desc.highest_order_diff_west_),
00102     highest_order_diff_east_(desc.highest_order_diff_east_),

```

```

00103     west_condition_(desc.west_condition_),
00104     east_condition_(desc.east_condition_) {}
00105
00106 mtk::RobinBCDescriptor1D::~RobinBCDescriptor1D() noexcept {}
00107
00108 int mtk::RobinBCDescriptor1D::highest_order_diff_west()
00109     const noexcept {
00110     return highest_order_diff_west_;
00111 }
00112
00113 int mtk::RobinBCDescriptor1D::highest_order_diff_east()
00114     const noexcept {
00115     return highest_order_diff_east_;
00116 }
00117
00118 void mtk::RobinBCDescriptor1D::PushBackWestCoeff(
00119     mtk::CoefficientFunction0D cw) {
00120
00121     #ifdef MTK_PERFORM_PREVENTIONS
00122     mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00123     mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00124         __FILE__, __LINE__, __func__);
00125     #endif
00126
00127     west_coefficients_.push_back(cw);
00128
00129     highest_order_diff_west_++;
00130 }
00131
00132 void mtk::RobinBCDescriptor1D::PushBackEastCoeff(
00133     mtk::CoefficientFunction0D ce) {
00134
00135     #ifdef MTK_PERFORM_PREVENTIONS
00136     mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00137     mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00138         __FILE__, __LINE__, __func__);
00139     #endif
00140
00141     east_coefficients_.push_back(ce);
00142
00143     highest_order_diff_east_++;
00144 }
00145
00146 void mtk::RobinBCDescriptor1D::set_west_condition(
00147     mtk::Real (*west_condition)(const mtk::Real &tt)) noexcept {
00148
00149     #ifdef MTK_PERFORM_PREVENTIONS
00150     mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00151     #endif
00152
00153     west_condition_ = west_condition;
00154 }
00155
00156 void mtk::RobinBCDescriptor1D::set_east_condition(
00157     mtk::Real (*east_condition)(const mtk::Real &tt)) noexcept {
00158
00159     #ifdef MTK_PERFORM_PREVENTIONS
00160     mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00161     #endif
00162
00163     east_condition_ = east_condition;
00164 }
00165
00166 bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix(
00167     const mtk::LaplD &lap,
00168     mtk::DenseMatrix &matrix,
00169     const mtk::Real &time) const {
00170
00171     #ifdef MTK_PERFORM_PREVENTIONS
00172     mtk::Tools::Prevent(highest_order_diff_west_ == -1,
00173         __FILE__, __LINE__, __func__);
00174     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00175         __FILE__, __LINE__, __func__);
00176     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00177     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00178     #endif
00179
00180     matrix.SetValue(0, 0, (west_coefficients_[0])(time));
00181
00182

```



```

00185 matrix.SetValue(matrix.num_rows() - 1,
00186                  matrix.num_cols() - 1,
00187                  (east_coefficients_[0])(time));
00188
00190 if (highest_order_diff_west_ > 0) {
00191
00193     mtk::Grad1D grad;
00194     if (!grad.ConstructGrad1D(lap.order_accuracy(),
00195                               lap.mimetic_threshold())) {
00196         return false;
00197     }
00198
00200     mtk::DenseMatrix coeffs(grad.mim_bndy());
00201
00203     mtk::Real idx = mtk::kOne/lap.delta();
00204
00206     for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00207         mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00208         mtk::Real unit_normal{~mtk::kOne};
00209         aux *= unit_normal*(west_coefficients_[1])(time);
00210         matrix.SetValue(0, ii, matrix.GetValue(0, ii) + aux);
00211     }
00212
00214     for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00215         mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00216         mtk::Real unit_normal{mtk::kOne};
00217         aux *= -unit_normal*(east_coefficients_[1])(time);
00218         matrix.SetValue(matrix.num_rows() - 1,
00219                         matrix.num_rows() - 1 - ii,
00220                         matrix.GetValue(matrix.num_rows() - 1,
00221                                         matrix.num_rows() - 1 - ii) + aux);
00222     }
00223 }
00224 return true;
00225 }
00226
00228 void mtk::RobinBCDescriptor1D::ImposeOnGrid(
00229     UniStgGrid1D &grid,
00230     const mtk::Real &time) const {
00231
00232     #ifdef MTK_PERFORM_PREVENTIONS
00233     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00234     mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00235     mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00236     #endif
00237
00238     (grid.discrete_field())[0] = west_condition_(time);
00239     (grid.discrete_field())[grid.num_cells_x() + 1] = east_condition_(time);
00240 }

```

18.113 src/mtk_robin_bc_descriptor_2d.cc File Reference

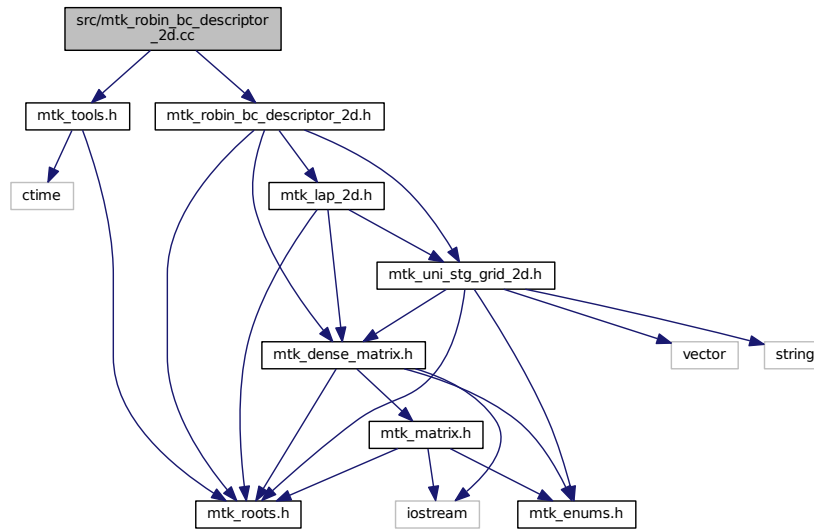
Impose Robin boundary conditions on the operators and on the grids.

```

#include "mtk_tools.h"
#include "mtk_robin_bc_descriptor_2d.h"

```

Include dependency graph for `mtk_robin_bc_descriptor_2d.cc`:



18.113.1 Detailed Description

This class presents an interface for the user to specify Robin boundary conditions on 2D mimetic operators and the grids they are acting on.

Def. Let $u(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ be the solution to an ordinary or partial differential equation of interest. We say that u satisfies a **Robin boundary condition** on $\partial\Omega$ if and only if there exists $\beta(\mathbf{x}, t) : \Omega \times [t_0, t_n] \mapsto \mathbb{R}$ so that:

$$\forall t \in [t_0, t_n] \forall \mathbf{x} \in \partial\Omega : \delta(\mathbf{x}, t)u(\mathbf{x}, t) + \eta(\mathbf{x}, t)(\hat{\mathbf{n}} \cdot \nabla u) = \beta(\mathbf{x}, t).$$

Intuitively, a **Robin boundary condition** is a constraint that must be satisfied by any linear combination of any scalar field u and its first normal derivative, in order for u to represent a unique solution to a given ordinary or partial differential equation of interest.

Instances of this class receive information about the coefficient functions and each condition for any subset of the boundary (west, east, south and north in 2D). These instances then handle the complexity of placing the coefficients in the differentiation matrices and the conditions in the grids.

See also

<http://mathworld.wolfram.com/NormalVector.html>

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_robin_bc_descriptor_2d.cc](#).

18.114 mtk_robin_bc_descriptor_2d.cc

```

00001
00034 /*
00035 Copyright (C) 2015, Computational Science Research Center, San Diego State
00036 University. All rights reserved.
00037
00038 Redistribution and use in source and binary forms, with or without modification,
00039 are permitted provided that the following conditions are met:
00040
00041 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00042 and a copy of the modified files should be reported once modifications are
00043 completed, unless these modifications are made through the project's GitHub
00044 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00045 should be developed and included in any deliverable.
00046
00047 2. Redistributions of source code must be done through direct
00048 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00049
00050 3. Redistributions in binary form must reproduce the above copyright notice,
00051 this list of conditions and the following disclaimer in the documentation and/or
00052 other materials provided with the distribution.
00053
00054 4. Usage of the binary form on proprietary applications shall require explicit
00055 prior written permission from the the copyright holders, and due credit should
00056 be given to the copyright holders.
00057
00058 5. Neither the name of the copyright holder nor the names of its contributors
00059 may be used to endorse or promote products derived from this software without
00060 specific prior written permission.
00061
00062 The copyright holders provide no reassurances that the source code provided does
00063 not infringe any patent, copyright, or any other intellectual property rights of
00064 third parties. The copyright holders disclaim any liability to any recipient for
00065 claims brought against recipient by any third party for infringement of that
00066 parties intellectual property rights.
00067
00068 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00069 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00070 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00071 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00072 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00073 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00074 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00075 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00076 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00077 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00078 */
00079
00080 #include "mtk_tools.h"
00081
00082 #include "mtk_robin_bc_descriptor_2d.h"
00083
00084 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D() :
00085     highest_order_diff_west_(-1),
00086     highest_order_diff_east_(-1),
00087     highest_order_diff_south_(-1),
00088     highest_order_diff_north_(-1),
00089     west_condition_(),
00090     east_condition_(),
00091     south_condition_(),
00092     north_condition_() {}
00093
00094 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D(
00095     const mtk::RobinBCDescriptor2D &desc):
00096     highest_order_diff_west_(desc.highest_order_diff_west_),
00097     highest_order_diff_east_(desc.highest_order_diff_east_),
00098     highest_order_diff_south_(desc.highest_order_diff_south_),
00099     highest_order_diff_north_(desc.highest_order_diff_north_),
00100     west_condition_(desc.west_condition_),
00101     east_condition_(desc.east_condition_),
00102     south_condition_(desc.south_condition_),
00103     north_condition_(desc.north_condition_) {}
00104
00105 mtk::RobinBCDescriptor2D::~~RobinBCDescriptor2D() noexcept {}
00106
00107 int mtk::RobinBCDescriptor2D::highest_order_diff_west()
00108     const noexcept {
00109     return highest_order_diff_west_;

```

```

00110 }
00111
00112 int mtk::RobinBCDescriptor2D::highest_order_diff_east()
    const noexcept {
00113
00114     return highest_order_diff_east_;
00115 }
00116
00117 int mtk::RobinBCDescriptor2D::highest_order_diff_south()
    const noexcept {
00118
00119     return highest_order_diff_south_;
00120 }
00121
00122 int mtk::RobinBCDescriptor2D::highest_order_diff_north()
    const noexcept {
00123
00124     return highest_order_diff_north_;
00125 }
00126
00127 void mtk::RobinBCDescriptor2D::PushBackWestCoeff(
    mtk::CoefficientFunction1D cw) {
00128
00129     #ifdef MTK_PERFORM_PREVENTIONS
00130     mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00131     mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00132                         __FILE__, __LINE__, __func__);
00133     #endif
00134
00135     west_coefficients_.push_back(cw);
00136     highest_order_diff_west_++;
00137 }
00138
00141 void mtk::RobinBCDescriptor2D::PushBackEastCoeff(
    mtk::CoefficientFunction1D ce) {
00142
00143     #ifdef MTK_PERFORM_PREVENTIONS
00144     mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00145     mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00146                         __FILE__, __LINE__, __func__);
00147     #endif
00148
00149     east_coefficients_.push_back(ce);
00150     highest_order_diff_east_++;
00151 }
00152
00155 void mtk::RobinBCDescriptor2D::PushBackSouthCoeff(
    mtk::CoefficientFunction1D cs) {
00156
00157     #ifdef MTK_PERFORM_PREVENTIONS
00158     mtk::Tools::Prevent(cs == nullptr, __FILE__, __LINE__, __func__);
00159     mtk::Tools::Prevent(highest_order_diff_south_ > 1,
00160                         __FILE__, __LINE__, __func__);
00161     #endif
00162
00163     south_coefficients_.push_back(cs);
00164     highest_order_diff_south_++;
00165 }
00166
00169 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff(
    mtk::CoefficientFunction1D cn) {
00170
00171     #ifdef MTK_PERFORM_PREVENTIONS
00172     mtk::Tools::Prevent(cn == nullptr, __FILE__, __LINE__, __func__);
00173     mtk::Tools::Prevent(highest_order_diff_north_ > 1,
00174                         __FILE__, __LINE__, __func__);
00175     #endif
00176
00177     north_coefficients_.push_back(cn);
00178     highest_order_diff_north_++;
00179 }
00180
00183 void mtk::RobinBCDescriptor2D::set_west_condition(
    mtk::Real (*west_condition)(const mtk::Real &yy,
                                const mtk::Real &tt)) noexcept {
00184
00185     #ifdef MTK_PERFORM_PREVENTIONS
00186
00187

```

```

00188     mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00189 #endif
00190
00191     west_condition_ = west_condition;
00192 }
00193
00194 void mtk::RobinBCDescriptor2D::set_east_condition(
00195     mtk::Real (*east_condition)(const mtk::Real &yy,
00196                                 const mtk::Real &tt)) noexcept {
00197
00198     #ifdef MTK_PERFORM_PREVENTIONS
00199     mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00200     #endif
00201
00202     east_condition_ = east_condition;
00203 }
00204
00205 void mtk::RobinBCDescriptor2D::set_south_condition(
00206     mtk::Real (*south_condition)(const mtk::Real &xx,
00207                                 const mtk::Real &tt)) noexcept {
00208
00209     #ifdef MTK_PERFORM_PREVENTIONS
00210     mtk::Tools::Prevent(south_condition == nullptr,
00211                         __FILE__, __LINE__, __func__);
00212     #endif
00213
00214     south_condition_ = south_condition;
00215 }
00216
00217 void mtk::RobinBCDescriptor2D::set_north_condition(
00218     mtk::Real (*north_condition)(const mtk::Real &xx,
00219                                 const mtk::Real &tt)) noexcept {
00220
00221     #ifdef MTK_PERFORM_PREVENTIONS
00222     mtk::Tools::Prevent(north_condition == nullptr,
00223                         __FILE__, __LINE__, __func__);
00224     #endif
00225
00226     north_condition_ = north_condition;
00227 }
00228
00229 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace
00230 (
00231     const mtk::Lap2D &lap,
00232     const mtk::UniStgGrid2D &grid,
00233     mtk::DenseMatrix &matrix,
00234     const mtk::Real &time) const {
00235
00236     // For the south-west corner:
00237     auto cc = (south_coefficients_[0])(grid.west_bndy(), time);
00238
00239     #if MTK_VERBOSE_LEVEL > 2
00240     std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00241         matrix.num_cols() << " columns." << std::endl;
00242     std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00243     #endif
00244
00245     matrix.SetValue(0, 0, cc);
00246
00247     // Compute first centers per dimension.
00248     auto first_center_x = grid.west_bndy() + grid.delta_x()/
00249         mtk::kTwo;
00250
00251     // For each entry on the diagonal (south boundary):
00252     for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00253         // Evaluate next set spatial coordinates to evaluate the coefficient.
00254         mtk::Real xx = first_center_x + ii*grid.delta_x();
00255         // Evaluate and assign the Dirichlet coefficient.
00256         cc = (south_coefficients_[0])(xx, time);
00257
00258         #if MTK_VERBOSE_LEVEL > 2
00259         std::cout << "Setting at " << ii + 1 << ' ' << ii + 1 << std::endl;
00260         #endif
00261
00262         matrix.SetValue(ii + 1, ii + 1, cc);
00263     }
00264
00265     // For the south-east corner:
00266     cc = (south_coefficients_[0])(grid.east_bndy(), time);
00267

```

```

00268 #if MTK_VERBOSE_LEVEL > 2
00269 std::cout << "Setting at " << grid.num_cells_x() + 1 << ' ' <<
00270     grid.num_cells_x() + 1 << std::endl;
00271 #endif
00272
00273 matrix.SetValue(grid.num_cells_x() + 1, grid.num_cells_x() + 1, cc);
00274
00275 if (highest_order_diff_south_ > 0) {
00276
00277 }
00280
00281 return true;
00282 }
00283
00284 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace
(
00285     const mtk::Lap2D &lap,
00286     const mtk::UniStgGrid2D &grid,
00287     mtk::DenseMatrix &matrix,
00288     const mtk::Real &time) const {
00289
00290
00291
00292
00293 // For each entry on the diagonal:
00294 for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00295     // Evaluate next set spatial coordinates to evaluate the coefficient.
00296     mtk::Real xx{(grid.discrete_domain_x())[ii]};
00297     // Evaluate and assign the Dirichlet coefficient.
00298     mtk::Real cc = (south_coefficients_[0])(xx, time);
00299     matrix.SetValue(ii, ii, cc);
00300 }
00301
00302 if (highest_order_diff_south_ > 0) {
00303
00304 }
00305
00306 return true;
00307 }
00308
00309 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace
(
00310     const mtk::Lap2D &lap,
00311     const mtk::UniStgGrid2D &grid,
00312     mtk::DenseMatrix &matrix,
00313     const mtk::Real &time) const {
00314
00315 int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00316
00317
00318 // For the north-west corner:
00319 mtk::Real cc =
00320     (north_coefficients_[0])(grid.west_bndy(), time);
00321
00322 #if MTK_VERBOSE_LEVEL > 2
00323 std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00324     matrix.num_cols() << " columns." << std::endl;
00325 std::cout << "Setting at " << north_offset << ' ' << north_offset <<
00326     std::endl;
00327 #endif
00328
00329 matrix.SetValue(north_offset, north_offset, cc);
00330
00331 // Compute first centers per dimension.
00332 auto first_center_x = grid.west_bndy() + grid.delta_x()/
mtk::kTwo;
00333
00334 // For each entry on the diagonal (north boundary):
00335 for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00336     // Evaluate next set spatial coordinates to evaluate the coefficient.
00337     mtk::Real xx = first_center_x + ii*grid.delta_x();
00338     // Evaluate and assign the Dirichlet coefficient.
00339     cc = (north_coefficients_[0])(xx, time);
00340
00341     #if MTK_VERBOSE_LEVEL > 2
00342     std::cout << "Setting at " << north_offset + ii + 1 << ' ' <<
00343         north_offset + ii + 1 << std::endl;
00344     #endif
00345
00346     matrix.SetValue(north_offset + ii + 1, north_offset + ii + 1, cc);
00347 }
00348
00349
00350
00351
00352

```

```

00353 // For the north-east corner:
00354 cc = (north_coefficients_[0])(grid.east_bndy(), time);
00355
00356 #if MTK_VERBOSE_LEVEL > 2
00357 std::cout << "Setting at " << north_offset + grid.num_cells_x() + 1 <<
00358     ' ' << north_offset + grid.num_cells_x() + 1 << std::endl;
00359 #endif
00360
00361 matrix.SetValue(north_offset + grid.num_cells_x() + 1,
00362     north_offset + grid.num_cells_x() + 1, cc);
00363
00364 if (highest_order_diff_north_ > 0) {
00365 }
00366
00367 return true;
00370 }
00371
00372 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace
00373 (
00374     const mtk::Lap2D &lap,
00375     const mtk::UniStgGrid2D &grid,
00376     mtk::DenseMatrix &matrix,
00377     const mtk::Real &time) const {
00378
00379     int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00380
00381     for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00382         mtk::Real xx{(grid.discrete_domain_x())[ii]};
00383         mtk::Real cc = (north_coefficients_[0])(xx, time);
00384         matrix.SetValue(north_offset + ii, north_offset + ii, cc);
00385     }
00386
00387     if (highest_order_diff_north_ > 0) {
00388     }
00389
00390     return true;
00391 }
00392
00393 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace
00394 (
00395     const mtk::Lap2D &lap,
00396     const mtk::UniStgGrid2D &grid,
00397     mtk::DenseMatrix &matrix,
00398     const mtk::Real &time) const {
00399
00400     // For the south-west corner:
00401     auto cc = (west_coefficients_[0])(grid.south_bndy(), time);
00402
00403     #if MTK_VERBOSE_LEVEL > 2
00404     std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00405         matrix.num_cols() << " columns." << std::endl;
00406     std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00407     #endif
00408
00409     mtk::Real harmonic_mean = mtk::kOne/matrix.GetValue(0, 0) +
00410         mtk::kOne/cc;
00411     harmonic_mean = mtk::kTwo/harmonic_mean;
00412     matrix.SetValue(0, 0, harmonic_mean);
00413
00414     int west_offset{grid.num_cells_x() + 1};
00415
00416     auto first_center_y = grid.south_bndy() + grid.delta_y()/
00417         mtk::kTwo;
00418
00419     // For each west entry on the diagonal (west boundary):
00420     for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00421         // Evaluate next set spatial coordinates to evaluate the coefficient.
00422         mtk::Real yy = first_center_y + ii*grid.delta_y();
00423         // Evaluate and assign the Dirichlet coefficient.
00424         cc = (west_coefficients_[0])(yy, time);
00425
00426         #if MTK_VERBOSE_LEVEL > 2
00427         std::cout << "Setting at " << west_offset + ii + 1 << ' ' <<
00428             west_offset + ii + 1 << std::endl;
00429         #endif
00430     }

```

```

00440
00441     matrix.SetValue(west_offset + ii + 1, west_offset + ii + 1, cc);
00442
00443     west_offset += grid.num_cells_x() + 1;
00444 }
00445
00446 // For the north-west corner:
00447 cc = (west_coefficients_[0])(grid.north_bndy(), time);
00448
00449 west_offset += grid.num_cells_x() + 1;
00450 int aux{west_offset};
00451 #if MTK_VERBOSE_LEVEL > 2
00452 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00453 #endif
00454
00455 harmonic_mean = mtk::kOne/matrix.GetValue(aux, aux) +
mtk::kOne/cc;
00456 harmonic_mean = mtk::kTwo/harmonic_mean;
00457
00458 matrix.SetValue(aux, aux, harmonic_mean);
00459
00460 if (highest_order_diff_west_ > 0) {
00461 }
00462
00463 return true;
00464 }
00465 }
00466 }
00467
00468 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace
(
00469     const mtk::Lap2D &lap,
00470     const mtk::UniStgGrid2D &grid,
00471     mtk::DenseMatrix &matrix,
00472     const mtk::Real &time) const {
00473
00474
00475
00476     int west_offset{grid.num_cells_x() + 1};
00477     // For each west entry on the diagonal:
00478     for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00479         // Evaluate next set spatial coordinates to evaluate the coefficient.
00480         mtk::Real yy{(grid.discrete_domain_y())[ii]};
00481         // Evaluate and assign the Dirichlet coefficient.
00482         mtk::Real cc = (west_coefficients_[0])(yy, time);
00483         matrix.SetValue(west_offset + ii, west_offset + ii, cc);
00484         west_offset += grid.num_cells_x() + 1;
00485     }
00486
00487     if (highest_order_diff_west_ > 0) {
00488     }
00489 }
00490
00491 return true;
00492 }
00493 }
00494
00495 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace
(
00496     const mtk::Lap2D &lap,
00497     const mtk::UniStgGrid2D &grid,
00498     mtk::DenseMatrix &matrix,
00499     const mtk::Real &time) const {
00500
00501
00502
00503     // For the south-east corner:
00504     auto cc = (east_coefficients_[0])(grid.south_bndy(), time);
00505
00506     int east_offset{grid.num_cells_x() + 1};
00507     #if MTK_VERBOSE_LEVEL > 2
00508     std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00509     matrix.num_cols() << " columns." << std::endl;
00510     std::cout << "Setting at " << east_offset << ' ' << east_offset <<
00511     std::endl;
00512     #endif
00513
00514     mtk::Real harmonic_mean =
00515     mtk::kOne/matrix.GetValue(east_offset, east_offset) +
mtk::kOne/cc;
00516     harmonic_mean = mtk::kTwo/harmonic_mean;
00517
00518     matrix.SetValue(east_offset, east_offset, harmonic_mean);
00519
00520     auto first_center_y = grid.south_bndy() + grid.delta_y()/

```



```

mtk::kTwo;
00521
00522 // For each east entry on the diagonal (east boundary):
00523 for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00524
00525     east_offset += grid.num_cells_x() + 1;
00526
00527     // Evaluate next set spatial coordinates to evaluate the coefficient.
00528     mtk::Real yy = first_center_y + ii*grid.delta_y();
00529     // Evaluate and assign the Dirichlet coefficient.
00530     cc = (east_coefficients_[0])(yy, time);
00531
00532     #if MTK_VERBOSE_LEVEL > 2
00533     std::cout << "Setting at " << east_offset + ii + 1 << ' ' <<
00534         east_offset + ii + 1 << std::endl;
00535     #endif
00536
00537     matrix.SetValue(east_offset + ii + 1, east_offset + ii + 1, cc);
00538 }
00539
00540 // For the north-east corner:
00541 cc = (east_coefficients_[0])(grid.north_bndy(), time);
00542
00543 east_offset += grid.num_cells_x() + 1;
00544 east_offset += grid.num_cells_x() + 1;
00545 int aux{east_offset};
00546 #if MTK_VERBOSE_LEVEL > 2
00547 std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00548 #endif
00549
00550 harmonic_mean =
00551     mtk::kOne/matrix.GetValue(aux, aux) + mtk::kOne/cc;
00552 harmonic_mean = mtk::kTwo/harmonic_mean;
00553
00554 matrix.SetValue(aux, aux, harmonic_mean);
00555
00556 if (highest_order_diff_east_ > 0) {
00557
00558 }
00559
00560 return true;
00561 }
00562 }
00563
00564 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace
(
00565     const mtk::Lap2D &lap,
00566     const mtk::UniStgGrid2D &grid,
00567     mtk::DenseMatrix &matrix,
00568     const mtk::Real &time) const {
00569
00570
00571
00572     int east_offset{grid.num_cells_x() + 1};
00573     // For each west entry on the diagonal:
00574     for (int ii = 0; ii < grid.num_cells_y() + 2; ++ii) {
00575         east_offset += grid.num_cells_x() + 1;
00576         // Evaluate next set spatial coordinates to evaluate the coefficient.
00577         mtk::Real yy{(grid.discrete_domain_y())[ii]};
00578         // Evaluate and assign the arithmetic mean of Dirichlet coefficients.
00579         mtk::Real cc = (east_coefficients_[0])(yy, time);
00580         matrix.SetValue(east_offset + ii, east_offset + ii, cc);
00581     }
00582
00583     if (highest_order_diff_east_ > 0) {
00584
00585     }
00586
00587     return true;
00588 }
00589 }
00590
00591 bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix(
00592     const mtk::Lap2D &lap,
00593     const mtk::UniStgGrid2D &grid,
00594     mtk::DenseMatrix &matrix,
00595     const mtk::Real &time) const {
00596
00597     #ifdef MTK_PERFORM_PREVENTIONS
00598     mtk::Tools::Prevent(highest_order_diff_south_ == -1,
00599         __FILE__, __LINE__, __func__);
00600     mtk::Tools::Prevent(highest_order_diff_north_ == -1,
00601         __FILE__, __LINE__, __func__);
00602     mtk::Tools::Prevent(highest_order_diff_west_ == -1,

```

```

00603         __FILE__, __LINE__, __func__);
00604     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00605         __FILE__, __LINE__, __func__);
00606     mtk::Tools::Prevent(grid.nature() !=
mtk::FieldNature::SCALAR,
00607         __FILE__, __LINE__, __func__);
00608     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00609     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00610     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00611     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00612     #endif
00613
00616
00617     bool success{true};
00618
00619     if (!grid.Bound()) {
00620         success = ImposeOnSouthBoundaryNoSpace(lap, grid, matrix, time);
00621         #ifdef MTK_PERFORM_PREVENTIONS
00622         if (!success) {
00623             return false;
00624         }
00625         #endif
00626         success = ImposeOnNorthBoundaryNoSpace(lap, grid, matrix, time);
00627         #ifdef MTK_PERFORM_PREVENTIONS
00628         if (!success) {
00629             return false;
00630         }
00631         #endif
00632         success = ImposeOnWestBoundaryNoSpace(lap, grid, matrix, time);
00633         #ifdef MTK_PERFORM_PREVENTIONS
00634         if (!success) {
00635             return false;
00636         }
00637         #endif
00638         success = ImposeOnEastBoundaryNoSpace(lap, grid, matrix, time);
00639         #ifdef MTK_PERFORM_PREVENTIONS
00640         if (!success) {
00641             return false;
00642         }
00643         #endif
00644     } else {
00645         success = ImposeOnSouthBoundaryWithSpace(lap, grid, matrix, time);
00646         #ifdef MTK_PERFORM_PREVENTIONS
00647         if (!success) {
00648             return false;
00649         }
00650         #endif
00651         success = ImposeOnNorthBoundaryWithSpace(lap, grid, matrix, time);
00652         #ifdef MTK_PERFORM_PREVENTIONS
00653         if (!success) {
00654             return false;
00655         }
00656         #endif
00657         success = ImposeOnWestBoundaryWithSpace(lap, grid, matrix, time);
00658         #ifdef MTK_PERFORM_PREVENTIONS
00659         if (!success) {
00660             return false;
00661         }
00662         #endif
00663         success = ImposeOnEastBoundaryWithSpace(lap, grid, matrix, time);
00664         #ifdef MTK_PERFORM_PREVENTIONS
00665         if (!success) {
00666             return false;
00667         }
00668         #endif
00669     }
00670
00671     return success;
00672 }
00673
00674 void mtk::RobinBCDescriptor2D::ImposeOnGrid(
00675     mtk::UniStgGrid2D &grid,
00676     const mtk::Real &time) const {
00677
00678     #ifdef MTK_PERFORM_PREVENTIONS
00679     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00680     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00681     mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00682     mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00683     mtk::Tools::Prevent(south_condition_ == nullptr,
00684         __FILE__, __LINE__, __func__);

```

```

00685     mtk::Tools::Prevent(north_condition_ == nullptr,
00686                         __FILE__, __LINE__, __func__);
00687     #endif
00688
00689     if (grid.nature() == mtk::FieldNature::SCALAR) {
00690
00691
00692
00693
00694         mtk::Real xx = grid.west_bndy();
00695         (grid.discrete_field())[0] = south_condition_(xx, time);
00696
00697         xx = xx + grid.delta_x()/mtk::kTwo;
00698         // For every point on the south boundary:
00699         for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00700             (grid.discrete_field())[ii + 1] =
00701                 south_condition_(xx + ii*grid.delta_x(), time);
00702         }
00703
00704         xx = grid.east_bndy();
00705         (grid.discrete_field())[grid.num_cells_x() + 1] =
00706             south_condition_(xx, time);
00707
00708
00709
00710
00711         xx = grid.west_bndy();
00712         int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00713         (grid.discrete_field())[north_offset] = north_condition_(xx, time);
00714
00715         xx = xx + grid.delta_x()/mtk::kTwo;
00716         for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00717             (grid.discrete_field())[north_offset + ii + 1] =
00718                 north_condition_(xx + ii*grid.delta_x(), time);
00719         }
00720
00721         xx = grid.east_bndy();
00722         (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00723             north_condition_(xx, time);
00724
00725
00726
00727
00728
00729
00730
00731
00732         mtk::Real yy = grid.south_bndy();
00733         (grid.discrete_field())[0] =
00734             ((grid.discrete_field())[0] + west_condition_(yy, time))/
00735             mtk::kTwo;
00736
00737         int west_offset{grid.num_cells_x() + 1 + 1};
00738         yy = yy + grid.delta_y()/mtk::kTwo;
00739         for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00740             #if MTK_VERBOSE_LEVEL > 2
00741                 std::cout << "Adding on " << west_offset << "-th position." << std::endl;
00742             #endif
00743             (grid.discrete_field())[west_offset] =
00744                 west_condition_(yy + ii*grid.delta_y(), time);
00745             west_offset += grid.num_cells_x() + 1 + 1;
00746         }
00747
00748         yy = grid.north_bndy();
00749         north_offset = (grid.num_cells_y() + 1)*(grid.num_cells_x() + 2);
00750         (grid.discrete_field())[north_offset] =
00751             ((grid.discrete_field())[north_offset] + west_condition_(yy, time))/
00752             mtk::kTwo;
00753
00754
00755
00756
00757
00758
00759
00760         yy = grid.south_bndy();
00761         int east_offset{grid.num_cells_x() + 1};
00762         (grid.discrete_field())[east_offset] =
00763             ((grid.discrete_field())[east_offset] + east_condition_(yy, time))/
00764             mtk::kTwo;
00765
00766
00767
00768
00769         yy = yy + grid.delta_y()/mtk::kTwo;
00770         for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00771             east_offset += grid.num_cells_x() + 1 + 1;
00772             #if MTK_VERBOSE_LEVEL > 2
00773                 std::cout << "Adding on " << east_offset << "-th position." << std::endl;
00774             #endif
00775             (grid.discrete_field())[east_offset] =
00776                 east_condition_(yy + ii*grid.delta_y(), time);
00777         }
00778
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
00786
00787
00788     }
00789 }
00790 }

```

18.115 src/mtk_tools.cc File Reference

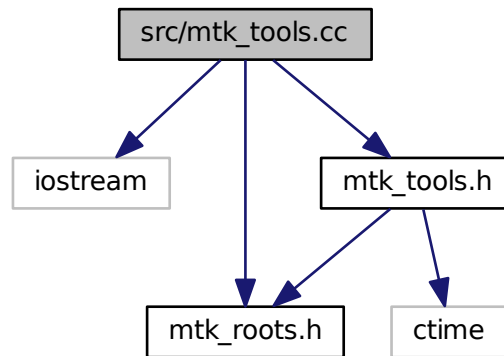
Tool manager class.

```

#include <iostream>
#include "mtk_roots.h"
#include "mtk_tools.h"

```

Include dependency graph for mtk_tools.cc:



18.115.1 Detailed Description

Implementation of a class providing basic tools to ensure execution correctness, and to assists with unitary testing.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_tools.cc](#).

18.116 mtk_tools.cc

```

00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017

```

```

00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <iostream>
00058
00059 #include "mtk_roots.h"
00060 #include "mtk_tools.h"
00061
00062 void mtk::Tools::Prevent(const bool condition,
00063                         const char *const fname,
00064                         int lineno,
00065                         const char *const fxname) noexcept {
00066
00067     if (lineno < 1) {
00068         std::cerr << __FILE__ << ": " << "Incorrect parameter at line " <<
00069         __LINE__ - 2 << " (" << __func__ << ")" << std::endl;
00070         exit(EXIT_FAILURE);
00071     }
00072
00073     if (condition) {
00074         std::cerr << fname << ": " << "Incorrect parameter at line " <<
00075         lineno << " (" << fxname << ")" << std::endl;
00076         exit(EXIT_FAILURE);
00077     }
00078 }
00079
00080 int mtk::Tools::test_number_{}; // Current test being executed.
00081
00082 mtk::Real mtk::Tools::duration_{}; // Duration of the current test.
00083
00084 clock_t mtk::Tools::begin_time_{}; // Elapsed time on current test.
00085
00086 void mtk::Tools::BeginUnitTestNo(const int &nn) noexcept {
00087
00088     #if MTK_PERFORM_PREVENTIONS
00089     mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00090     #endif
00091
00092     test_number_ = nn;
00093
00094     std::cout << "Beginning test " << nn << "." << std::endl;
00095     begin_time_ = clock();
00096 }
00097
00098 void mtk::Tools::EndUnitTestNo(const int &nn) noexcept {

```

```

00100
00101  #if MTK_PERFORM_PREVENTIONS
00102  mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00103  #endif
00104
00105  duration_ = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00106 }
00107
00108 void mtk::Tools::Assert(const bool &condition) noexcept {
00109
00110  if (condition) {
00111      std::cout << "Test " << test_number_ << ": PASSED in " << duration_ <<
00112      " s." << std::endl;
00113  } else {
00114      std::cout << "Test " << test_number_ << ": FAILED in " << duration_ <<
00115      " s." << std::endl;
00116  }
00117 }

```

18.117 src/mtk_uni_stg_grid_1d.cc File Reference

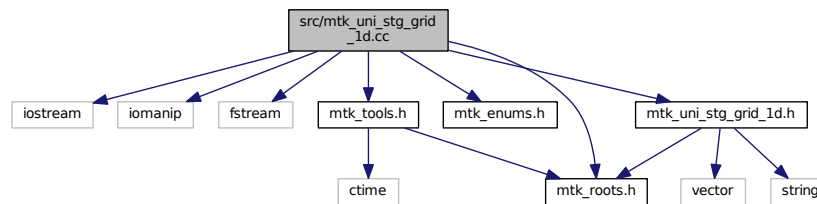
Implementation of an 1D uniform staggered grid.

```

#include <iostream>
#include <iomanip>
#include <fstream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_1d.h"

```

Include dependency graph for mtk_uni_stg_grid_1d.cc:



Namespaces

- [mtk](#)
Mimetic Methods Toolkit namespace.

Functions

- `std::ostream & mtk::operator<< (std::ostream &stream, mtk::UniStgGrid1D &in)`

18.117.1 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](#).

18.118 mtk_uni_stg_grid_1d.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_enums.h"
00062 #include "mtk_tools.h"
00063
00064 #include "mtk_uni_stg_grid_1d.h"
00065
00066 namespace mtk {
00067
00068 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid1D &in) {
00069
00070     stream << '[' << in.west_bndy_x << ':' << in.num_cells_x << ':' <<
00071     in.east_bndy_x << "]" = " << std::endl << std::endl;
00072
00073
00074
00075     stream << "x:";
00076     for (unsigned int ii = 0; ii < in.discrete_domain_x.size(); ++ii) {
00077         stream << std::setw(10) << in.discrete_domain_x[ii];

```

```

00078     }
00079     stream << std::endl;
00080
00082
00083     if (in.nature_ == mtk::FieldNature::SCALAR) {
00084         stream << "u:";
00085     }
00086     else {
00087         stream << "v:";
00088     }
00089     for (unsigned int ii = 0; ii < in.discrete_field_.size(); ++ii) {
00090         stream << std::setw(10) << in.discrete_field_[ii];
00091     }
00092
00093     stream << std::endl;
00094
00095     return stream;
00096 }
00097 }
00098
00099 mtk::UniStgGrid1D::UniStgGrid1D():
00100     nature_(),
00101     discrete_domain_x_(),
00102     discrete_field_(),
00103     west_bndy_x_(),
00104     east_bndy_x_(),
00105     num_cells_x_(),
00106     delta_x_() {}
00107
00108 mtk::UniStgGrid1D::UniStgGrid1D(const
UniStgGrid1D &grid):
00109     nature_(grid.nature_),
00110     west_bndy_x_(grid.west_bndy_x_),
00111     east_bndy_x_(grid.east_bndy_x_),
00112     num_cells_x_(grid.num_cells_x_),
00113     delta_x_(grid.delta_x_) {}
00114
00115     std::copy(grid.discrete_domain_x_.begin(),
00116             grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
00117             discrete_domain_x_.begin());
00118
00119     std::copy(grid.discrete_field_.begin(),
00120             grid.discrete_field_.begin() + grid.discrete_field_.size(),
00121             discrete_field_.begin());
00122 }
00123
00124 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00125                                 const Real &east_bndy_x,
00126                                 const int &num_cells_x,
00127                                 const mtk::FieldNature &nature) {
00128
00129     #ifdef MTK_PERFORM_PREVENTIONS
00130     mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00131     mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00132     mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00133     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00134     #endif
00135
00136     nature_ = nature;
00137     west_bndy_x_ = west_bndy_x;
00138     east_bndy_x_ = east_bndy_x;
00139     num_cells_x_ = num_cells_x;
00140
00141     delta_x_ = (east_bndy_x - west_bndy_x)/((mtk::Real) num_cells_x);
00142 }
00143
00144 mtk::UniStgGrid1D::~UniStgGrid1D() {}
00145
00146 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00147
00148     return west_bndy_x_;
00149 }
00150
00151 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00152
00153     return east_bndy_x_;
00154 }
00155
00156 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00157

```



```

00158     return delta_x_;
00159 }
00160
00161 const mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() const
00162 {
00163     return discrete_domain_x_.data();
00164 }
00165
00166 mtk::Real *mtk::UniStgGrid1D::discrete_field() {
00167     return discrete_field_.data();
00168 }
00169 }
00170
00171 int mtk::UniStgGrid1D::num_cells_x() const {
00172     return num_cells_x_;
00173 }
00174 }
00175
00176 void mtk::UniStgGrid1D::BindScalarField(
00177     mtk::Real (*ScalarField)(const mtk::Real &xx)) {
00178
00179     #ifdef MTK_PERFORM_PREVENTIONS
00180     mtk::Tools::Prevent(nature_ == mtk::FieldNature::VECTOR,
00181         __FILE__, __LINE__, __func__);
00182     #endif
00183
00184     discrete_domain_x_.reserve(num_cells_x_ + 2);
00185
00186     discrete_domain_x_.push_back(west_bndy_x_);
00187     #ifdef MTK_PRECISION_DOUBLE
00188     auto first_center = west_bndy_x_ + delta_x_/2.0;
00189     #else
00190     auto first_center = west_bndy_x_ + delta_x_/2.0f;
00191     #endif
00192     discrete_domain_x_.push_back(first_center);
00193     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00194         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00195     }
00196     discrete_domain_x_.push_back(east_bndy_x_);
00197
00198     discrete_field_.reserve(num_cells_x_ + 2);
00199
00200     discrete_field_.push_back(ScalarField(west_bndy_x_));
00201
00202     discrete_field_.push_back(ScalarField(first_center));
00203     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00204         discrete_field_.push_back(ScalarField(first_center + ii*delta_x_));
00205     }
00206     discrete_field_.push_back(ScalarField(east_bndy_x_));
00207 }
00208
00209 void mtk::UniStgGrid1D::BindVectorField(
00210     mtk::Real (*VectorField)(mtk::Real xx)) {
00211
00212     #ifdef MTK_PERFORM_PREVENTIONS
00213     mtk::Tools::Prevent(nature_ == mtk::FieldNature::SCALAR,
00214         __FILE__, __LINE__,
00215         __func__);
00216     #endif
00217
00218     discrete_domain_x_.reserve(num_cells_x_ + 1);
00219
00220     discrete_domain_x_.push_back(west_bndy_x_);
00221     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00222         discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00223     }
00224     discrete_domain_x_.push_back(east_bndy_x_);
00225
00226     discrete_field_.reserve(num_cells_x_ + 1);
00227
00228     discrete_field_.push_back(VectorField(west_bndy_x_));
00229     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00230         discrete_field_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00231     }
00232     discrete_field_.push_back(VectorField(east_bndy_x_));
00233 }
00234 }

```

```

00241
00242 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00243                                     std::string space_name,
00244                                     std::string field_name) const {
00245
00246     std::ofstream output_dat_file; // Output file.
00247
00248     output_dat_file.open(filename);
00249
00250     if (!output_dat_file.is_open()) {
00251         return false;
00252     }
00253
00254     output_dat_file << "#" << space_name << " " << field_name << std::endl;
00255     for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00256         output_dat_file << discrete_domain_x_[ii] << " " << discrete_field_[ii] <<
00257             std::endl;
00258     }
00259
00260     output_dat_file.close();
00261
00262     return true;
00263 }

```

18.119 src/mtk_uni_stg_grid_2d.cc File Reference

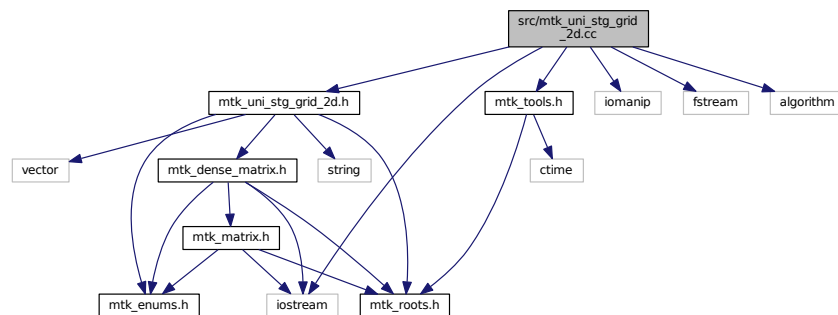
Implementation of a 2D uniform staggered grid.

```

#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)`

18.119.1 Detailed Description

Implementation of a 2D uniform staggered grid.

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_2d.cc](#).

18.120 mtk_uni_stg_grid_2d.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_2d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid2D &in) {
00068
00069     stream << '[' << in.west_bndy_ << ':' << in.num_cells_x << ':' <<

```

```

00070     in.east_bndy_ << "]" x ";
00071
00072     stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073     in.north_bndy_ << "]" = " << std::endl << std::endl;
00074
00075
00076
00077     stream << "x:";
00078     for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00079         stream << std::setw(10) << in.discrete_domain_x_[ii];
00080     }
00081     stream << std::endl;
00082
00083     stream << "y:";
00084     for (unsigned int ii = 0; ii < in.discrete_domain_y_.size(); ++ii) {
00085         stream << std::setw(10) << in.discrete_domain_y_[ii];
00086     }
00087     stream << std::endl;
00088
00089
00090
00091     if (in.nature_ == mtk::FieldNature::SCALAR) {
00092         stream << "u:" << std::endl;
00093         if (in.discrete_field_.size() > 0) {
00094             for (int ii = 0; ii < in.num_cells_x_ + 2; ++ii) {
00095                 for (int jj = 0; jj < in.num_cells_y_ + 2; ++jj) {
00096                     stream << std::setw(10) << in.discrete_field_[ii*in.
num_cells_y_ +
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
UniStgGrid2D &grid):
00147     nature_(grid.nature_),
00148     west_bndy_(grid.west_bndy_),
00149     east_bndy_(grid.east_bndy_),
00150     num_cells_x_(grid.num_cells_x_),

```

```

00151     delta_x_(grid.delta_x_),
00152     south_bndy_(grid.south_bndy_),
00153     north_bndy_(grid.north_bndy_),
00154     num_cells_y_(grid.num_cells_y_),
00155     delta_y_(grid.delta_y_) {
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_field_.begin(),
00168               grid.discrete_field_.begin() + grid.discrete_field_.size(),
00169               discrete_field_.begin());
00170 }
00171
00172 mtk::UniStgGrid2D::UniStgGrid2D(const Real &west_bndy,
00173                                 const Real &east_bndy,
00174                                 const int &num_cells_x,
00175                                 const Real &south_bndy,
00176                                 const Real &north_bndy,
00177                                 const int &num_cells_y,
00178                                 const mtk::FieldNature &nature) {
00179
00180 #ifdef MTK_PERFORM_PREVENTIONS
00181     mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00182     mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00183     mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00184     mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00185     mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00186     mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00187     mtk::Tools::Prevent(north_bndy <= south_bndy,
00188                         __FILE__, __LINE__, __func__);
00189     mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00190 #endif
00191
00192     nature_ = nature;
00193
00194     west_bndy_ = west_bndy;
00195     east_bndy_ = east_bndy;
00196     num_cells_x_ = num_cells_x;
00197
00198     south_bndy_ = south_bndy;
00199     north_bndy_ = north_bndy;
00200     num_cells_y_ = num_cells_y;
00201
00202     delta_x_ = (east_bndy_ - west_bndy_) / (mtk::Real) num_cells_x;
00203     delta_y_ = (north_bndy_ - south_bndy_) / (mtk::Real) num_cells_y;
00204 }
00205
00206 mtk::UniStgGrid2D::~UniStgGrid2D() {}
00207
00208 mtk::FieldNature mtk::UniStgGrid2D::nature() const {
00209     return nature_;
00210 }
00211
00212 mtk::Real mtk::UniStgGrid2D::west_bndy() const {
00213     return west_bndy_;
00214 }
00215
00216 mtk::Real mtk::UniStgGrid2D::east_bndy() const {
00217     return east_bndy_;
00218 }
00219
00220 int mtk::UniStgGrid2D::num_cells_x() const {
00221     return num_cells_x_;
00222 }
00223
00224 mtk::Real mtk::UniStgGrid2D::delta_x() const {
00225     return delta_x_;
00226 }
00227
00228 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00229     return delta_y_;
00230 }

```

```

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
00238     return south_bndy_;
00239 }
00240
00241 mtk::Real mtk::UniStgGrid2D::north_bndy() const {
00242
00243     return north_bndy_;
00244 }
00245
00246 int mtk::UniStgGrid2D::num_cells_y() const {
00247
00248     return num_cells_y_;
00249 }
00250
00251 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00252
00253     return delta_y_;
00254 }
00255
00256 bool mtk::UniStgGrid2D::Bound() const {
00257
00258     return discrete_field_.size() != 0;
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     discrete_domain_x_.reserve(num_cells_x_ + 2);
00286
00287     discrete_domain_x_.push_back(west_bndy_);
00288     #ifdef MTK_PRECISION_DOUBLE
00289     auto first_center = west_bndy_ + delta_x_/2.0;
00290     #else
00291     auto first_center = west_bndy_ + delta_x_/2.0f;
00292     #endif
00293     discrete_domain_x_.push_back(first_center);
00294     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00295         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00296     }
00297     discrete_domain_x_.push_back(east_bndy_);
00298
00299
00300     discrete_domain_y_.reserve(num_cells_y_ + 2);
00301
00302     discrete_domain_y_.push_back(south_bndy_);
00303     #ifdef MTK_PRECISION_DOUBLE
00304     first_center = south_bndy_ + delta_x_/2.0;
00305     #else
00306     first_center = south_bndy_ + delta_x_/2.0f;
00307     #endif
00308     discrete_domain_y_.push_back(first_center);
00309     for (auto ii = 1; ii < num_cells_y_; ++ii) {
00310         discrete_domain_y_.push_back(first_center + ii*delta_y_);
00311     }
00312     discrete_domain_y_.push_back(north_bndy_);
00313 }

```

```

00310 discrete_domain_y_.push_back(first_center);
00311 for (auto ii = 1; ii < num_cells_y_; ++ii) {
00312     discrete_domain_y_.push_back(first_center + ii*delta_y_);
00313 }
00314 discrete_domain_y_.push_back(north_bndy_);
00315
00317 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
00349
00350     // We need every data point of the discrete domain; i.e. we need all the
00351     // nodes and all the centers. There are mm centers for the x direction, and
00352     // nn centers for the y direction. Since there is one node per center, that
00353     // amounts to 2*mm. If we finally consider the final boundary node, it
00354     // amounts to a total of 2*mm + 1 for the x direction. Analogously, for the
00355     // y direction, this amounts to 2*nn + 1.
00356
00357     discrete_domain_x_.reserve(2*mm + 1);
00358
00359     discrete_domain_x_.push_back(west_bndy_);
00360     for (int ii = 1; ii < (2*mm + 1); ++ii) {
00361         discrete_domain_x_.push_back(west_bndy_ + ii*half_delta_x);
00362     }
00363
00365
00366     discrete_domain_y_.reserve(2*nn + 1);
00367
00368     discrete_domain_y_.push_back(south_bndy_);
00369     for (int ii = 1; ii < (2*nn + 1); ++ii) {
00370         discrete_domain_y_.push_back(south_bndy_ + ii*half_delta_y);
00371     }
00372
00374     discrete_field_.reserve(total);
00376
00377     // For each y-center.
00378     for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00379
00380         // Bind all of the x-nodes for this y-center.
00381         for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00382             discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00383                                                 discrete_domain_y_[ii]));
00384
00385             #if MTK_VERBOSE_LEVEL > 6
00386             std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00387                 discrete_domain_y_[ii] << " = " <<
00388                 VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00389             #endif
00390         }
00391     }
00392
00393     #if MTK_VERBOSE_LEVEL > 6
00394     std::cout << std::endl;
00395

```

```

00394 #endif
00395 }
00396
00397 void mtk::UniStgGrid2D::BindVectorFieldQComponent(
00398     mtk::Real (*VectorField)(const mtk::Real &xx, const
00399     mtk::Real &yy)) {
00399
00400     int mm{num_cells_x_};
00401     int nn{num_cells_y_};
00402
00403     // For each y-node.
00404     for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00405         // Bind all of the x-center for this y-node.
00406         for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00407             discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00408             discrete_domain_y_[ii]));
00409
00410             #if MTK_VERBOSE_LEVEL > 6
00411             std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00412             discrete_domain_y_[ii] << " = " <<
00413             VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00414             #endif
00415         }
00416     }
00417
00418     #if MTK_VERBOSE_LEVEL > 6
00419     std::cout << std::endl;
00420     #endif
00421 }
00422
00423 void mtk::UniStgGrid2D::BindVectorField(
00424     Real (*VectorFieldPComponent)(const Real &xx, const Real &yy),
00425     Real (*VectorFieldQComponent)(const Real &xx, const Real &yy)) {
00426
00427     #ifdef MTK_PERFORM_PREVENTIONS
00428     mtk::Tools::Prevent(nature_ != mtk::FieldNature::VECTOR,
00429     __FILE__, __LINE__,
00430     __func__);
00431     #endif
00432
00433     BindVectorFieldPComponent(VectorFieldPComponent);
00434     BindVectorFieldQComponent(VectorFieldQComponent);
00435 }
00436
00437 bool mtk::UniStgGrid2D::WriteToFile(std::string filename,
00438     std::string space_name_x,
00439     std::string space_name_y,
00440     std::string field_name) const {
00441
00442     std::ofstream output_dat_file; // Output file.
00443
00444     output_dat_file.open(filename);
00445
00446     if (!output_dat_file.is_open()) {
00447         return false;
00448     }
00449
00450     if (nature_ == mtk::FieldNature::SCALAR) {
00451         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00452         field_name << std::endl;
00453
00454         int idx{};
00455         for (unsigned int ii = 0; ii < discrete_domain_y_.size(); ++ii) {
00456             for (unsigned int jj = 0; jj < discrete_domain_x_.size(); ++jj) {
00457                 output_dat_file << discrete_domain_x_[jj] << ' ' <<
00458                 discrete_domain_y_[ii] << ' ' <<
00459                 discrete_field_[idx] <<
00460                 std::endl;
00461                 idx++;
00462             }
00463             output_dat_file << std::endl;
00464         }
00465     } else {
00466         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00467         field_name << std::endl;
00468
00469         output_dat_file << "# Horizontal component:" << std::endl;
00470
00471         int mm{num_cells_x_};
00472         int nn{num_cells_y_};
00473     }

```



```

00474
00475
00476 // For each y-center.
00477 int idx{};
00478 for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00479     // Bind all of the x-nodes for this y-center.
00480     for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00481         output_dat_file << discrete_domain_x[jj] << ' ' <<
00482             discrete_domain_y[ii] << ' ' << discrete_field[idx] << ' ' <<
00483             mtk::kZero << std::endl;
00484         ++idx;
00485     }
00486 }
00487
00488 int p_offset{nn*(mm + 1) - 1};
00489 idx = 0;
00490 output_dat_file << "# Vertical component:" << std::endl;
00491 // For each y-node.
00492 for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00493     // Bind all of the x-center for this y-node.
00494     for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00495         output_dat_file << discrete_domain_x[jj] << ' ' <<
00496             discrete_domain_y[ii] << ' ' << mtk::kZero << ' ' <<
00497             discrete_field[p_offset + idx] << std::endl;
00498         ++idx;
00499     }
00500 }
00501
00502 output_dat_file.close();
00503
00504 return true;
00505 }

```

18.121 src/mtk_uni_stg_grid_3d.cc File Reference

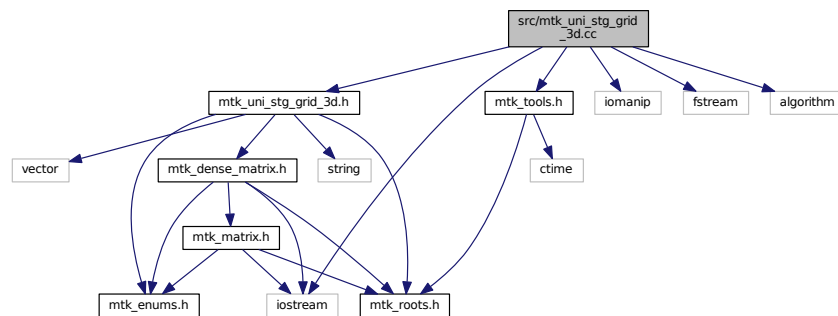
Implementation of a 3D uniform staggered grid.

```

#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)`

18.121.1 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](#).

18.122 [mtk_uni_stg_grid_3d.cc](#)

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```

```

00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid3D &in) {
00068
00069     stream << '[' << in.west_bndy_ << ':' << in.num_cells_x_ << ':' <<
00070     in.east_bndy_ << "] x ";
00071
00072     stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073     in.north_bndy_ << "] y ";
00074
00075     stream << '[' << in.bottom_bndy_ << ':' << in.num_cells_z_ << ':' <<
00076     in.top_bndy_ << "] = " << std::endl << std::endl;
00077
00078     stream << "x:";
00079     for (auto const &cc: in.discrete_domain_x_) {
00080         stream << std::setw(10) << cc;
00081     }
00082     stream << std::endl;
00083
00084     stream << "y:";
00085     for (auto const &cc: in.discrete_domain_y_) {
00086         stream << std::setw(10) << cc;
00087     }
00088     stream << std::endl;
00089
00090     stream << "z:";
00091     for (auto const &cc: in.discrete_domain_z_) {
00092         stream << std::setw(10) << cc;
00093     }
00094     stream << std::endl;
00095
00096     if (in.nature_ == mtk::FieldNature::SCALAR) {
00097         stream << "u(x,y,z):" << std::endl;
00098         if (in.discrete_field_.size() > 0) {
00099
00100         }
00101     } else {
00102         stream << "p(x,y,z):" << std::endl;
00103         stream << "q(x,y,z):" << std::endl;
00104         if (in.discrete_field_.size() > 0) {
00105
00106         }
00107     }
00108     return stream;
00109 }
00110
00111 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator=(const
00112 mtk::UniStgGrid3D &in) {
00113
00114     UniStgGrid3D out(in);
00115
00116     return out;
00117 }
00118
00119 mtk::UniStgGrid3D::UniStgGrid3D():
00120     discrete_domain_x_(),
00121     discrete_domain_y_(),
00122     discrete_domain_z_(),
00123     discrete_field_(),
00124     nature_(),
00125     west_bndy_(),
00126     east_bndy_(),
00127     num_cells_x_(),
00128     delta_x_(),
00129     south_bndy_(),
00130     north_bndy_(),
00131     num_cells_y_(),

```

```

00136     delta_y_(),
00137     bottom_bndy_(),
00138     top_bndy_(),
00139     num_cells_z_(),
00140     delta_z_() {}
00141
00142 mtk::UniStgGrid3D::UniStgGrid3D(const
UniStgGrid3D &grid):
00143     nature_(grid.nature_),
00144     west_bndy_(grid.west_bndy_),
00145     east_bndy_(grid.east_bndy_),
00146     num_cells_x_(grid.num_cells_x_),
00147     delta_x_(grid.delta_x_),
00148     south_bndy_(grid.south_bndy_),
00149     north_bndy_(grid.north_bndy_),
00150     num_cells_y_(grid.num_cells_y_),
00151     delta_y_(grid.delta_y_),
00152     bottom_bndy_(grid.bottom_bndy_),
00153     top_bndy_(grid.top_bndy_),
00154     num_cells_z_(grid.num_cells_z_),
00155     delta_z_(grid.delta_z_) {
00156
00157     std::copy(grid.discrete_domain_x_.begin(),
00158               grid.discrete_domain_x_.begin() + grid.
discrete_domain_x_.size(),
00159               discrete_domain_x_.begin());
00160
00161     std::copy(grid.discrete_domain_y_.begin(),
00162               grid.discrete_domain_y_.begin() + grid.
discrete_domain_y_.size(),
00163               discrete_domain_y_.begin());
00164
00165     std::copy(grid.discrete_domain_z_.begin(),
00166               grid.discrete_domain_z_.begin() + grid.
discrete_domain_z_.size(),
00167               discrete_domain_z_.begin());
00168
00169     std::copy(grid.discrete_field_.begin(),
00170               grid.discrete_field_.begin() + grid.discrete_field_.size(),
00171               discrete_field_.begin());
00172 }
00173
00174 mtk::UniStgGrid3D::UniStgGrid3D(const Real &west_bndy,
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
00225     return nature_;
00226 }
00227
00228 mtk::Real mtk::UniStgGrid3D::west_bndy() const {
00229
00230     return west_bndy_;
00231 }
00232
00233 mtk::Real mtk::UniStgGrid3D::east_bndy() const {
00234
00235     return east_bndy_;
00236 }
00237
00238 int mtk::UniStgGrid3D::num_cells_x() const {
00239
00240     return num_cells_x_;
00241 }
00242
00243 mtk::Real mtk::UniStgGrid3D::delta_x() const {
00244
00245     return delta_x_;
00246 }
00247
00248 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_x() const
00249 {
00250     return discrete_domain_x_.data();
00251 }
00252
00253 mtk::Real mtk::UniStgGrid3D::south_bndy() const {
00254
00255     return south_bndy_;
00256 }
00257
00258 mtk::Real mtk::UniStgGrid3D::north_bndy() const {
00259
00260     return north_bndy_;
00261 }
00262
00263 int mtk::UniStgGrid3D::num_cells_y() const {
00264
00265     return num_cells_y_;
00266 }
00267
00268 mtk::Real mtk::UniStgGrid3D::delta_y() const {
00269
00270     return delta_y_;
00271 }
00272
00273 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_y() const
00274 {
00275     return discrete_domain_y_.data();
00276 }
00277
00278 mtk::Real mtk::UniStgGrid3D::bottom_bndy() const {
00279
00280     return bottom_bndy_;
00281 }
00282
00283 mtk::Real mtk::UniStgGrid3D::top_bndy() const {
00284
00285     return top_bndy_;
00286 }
00287
00288 int mtk::UniStgGrid3D::num_cells_z() const {
00289
00290     return num_cells_z_;
00291 }

```

```

00292
00293 mtk::Real mtk::UniStgGrid3D::delta_z() const {
00294
00295     return delta_z_;
00296 }
00297
00298 const mtk::Real* mtk::UniStgGrid3D::discrete_domain_z() const
00299 {
00300     return discrete_domain_z_.data();
00301 }
00302
00303 mtk::Real* mtk::UniStgGrid3D::discrete_field() {
00304
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
00330     discrete_domain_x_.reserve(num_cells_x_ + 2);
00331
00332     discrete_domain_x_.push_back(west_bndy_);
00333     #ifdef MTK_PRECISION_DOUBLE
00334     auto first_center = west_bndy_ + delta_x_/2.0;
00335     #else
00336     auto first_center = west_bndy_ + delta_x_/2.0f;
00337     #endif
00338     discrete_domain_x_.push_back(first_center);
00339     for (auto ii = 1; ii < num_cells_x_; ++ii) {
00340         discrete_domain_x_.push_back(first_center + ii*delta_x_);
00341     }
00342     discrete_domain_x_.push_back(east_bndy_);
00343
00344
00345     discrete_domain_y_.reserve(num_cells_y_ + 2);
00346
00347     discrete_domain_y_.push_back(south_bndy_);
00348     #ifdef MTK_PRECISION_DOUBLE
00349     first_center = south_bndy_ + delta_x_/2.0;
00350     #else
00351     first_center = south_bndy_ + delta_x_/2.0f;
00352     #endif
00353     discrete_domain_y_.push_back(first_center);
00354     for (auto ii = 1; ii < num_cells_y_; ++ii) {
00355         discrete_domain_y_.push_back(first_center + ii*delta_y_);
00356     }
00357     discrete_domain_y_.push_back(north_bndy_);
00358
00359
00360     discrete_domain_z_.reserve(num_cells_z_ + 2);
00361
00362     discrete_domain_z_.push_back(bottom_bndy_);
00363     first_center = bottom_bndy_ + delta_z_/mtk::kTwo;
00364     discrete_domain_z_.push_back(first_center);
00365     for (auto ii = 1; ii < num_cells_z_; ++ii) {
00366         discrete_domain_z_.push_back(first_center + ii*delta_z_);
00367     }
00368     discrete_domain_z_.push_back(top_bndy_);
00369
00370
00371
00372
00373     int aux{(num_cells_x_ + 2)*(num_cells_y_ + 2)*(num_cells_z_ + 2)};
00374

```

```

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                 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 mtk::Real &xx,
00417                                           const mtk::Real &yy,
00418                                           const mtk::Real &zz),
00419     mtk::Real (*VectorFieldQComponent) (const mtk::Real &xx,
00420                                           const mtk::Real &yy,
00421                                           const mtk::Real &zz),
00422     mtk::Real (*VectorFieldRComponent) (const mtk::Real &xx,
00423                                           const mtk::Real &yy,
00424                                           const mtk::Real &zz)) {
00425
00426     #ifdef MTK_PERFORM_PREVENTIONS
00427     mtk::Tools::Prevent(nature_ != mtk::FieldNature::VECTOR,
00428         __FILE__, __LINE__,
00429         __func__);
00430     #endif
00431
00432     BindVectorFieldPComponent(VectorFieldPComponent);
00433     BindVectorFieldQComponent(VectorFieldQComponent);
00434 }
00435
00436 bool mtk::UniStgGrid3D::WriteToFile(std::string filename,
00437                                     std::string space_name_x,
00438                                     std::string space_name_y,
00439                                     std::string space_name_z,
00440                                     std::string field_name) const {
00441
00442     std::ofstream output_dat_file; // Output file.
00443
00444     output_dat_file.open(filename);
00445
00446     if (!output_dat_file.is_open()) {
00447         return false;
00448     }
00449
00450     if (nature_ == mtk::FieldNature::SCALAR) {
00451         output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00452         space_name_z << ' ' << field_name << std::endl;
00453
00454         int idx{};
00455         for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {

```

```

00455     for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00456         for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00457             output_dat_file << discrete_domain_x_[jj] << ' ' <<
00458                 discrete_domain_y_[ii] << ' ' << discrete_domain_z_[kk] << ' ' <<
00459                 discrete_field_[idx] << std::endl;
00460             idx++;
00461         }
00462     }
00463 }
00464
00465 } else {
00466     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00467         space_name_z << ' ' << field_name << std::endl;
00468 }
00469 }
00470
00471 output_dat_file.close();
00472
00473 return true;
00474 }

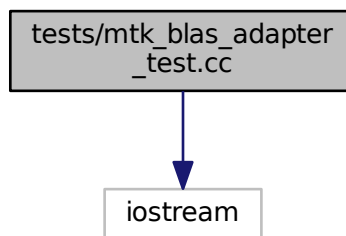
```

18.123 tests/mtk_blas_adapter_test.cc File Reference

Test file for the [mtk::BLASAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_blas_adapter_test.cc`:



Functions

- `int main ()`

18.123.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_blas_adapter_test.cc](#).

18.123.2 Function Documentation

18.123.2.1 int main ()

Definition at line 109 of file [mtk_blas_adapter_test.cc](#).

18.124 mtk_blas_adapter_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestRealDenseMM() {
00061
00062     mtk::Tools::BeginUnitTestNo(1);
00063
00064     int rr = 2;
00065     int cc = 3;
00066
00067     mtk::DenseMatrix aa(rr,cc);
00068
00069     aa.SetValue(0,0,1.0);
00070     aa.SetValue(0,1,2.0);
00071     aa.SetValue(0,2,3.0);
00072     aa.SetValue(1,0,4.0);
00073     aa.SetValue(1,1,5.0);
00074     aa.SetValue(1,2,6.0);
00075
00076     mtk::DenseMatrix bb(cc,rr);

```

```

00077
00078  bb.SetValue(0,0,7.0);
00079  bb.SetValue(0,1,8.0);
00080  bb.SetValue(1,0,9.0);
00081  bb.SetValue(1,1,10.0);
00082  bb.SetValue(2,0,11.0);
00083  bb.SetValue(2,1,12.0);
00084
00085  mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00086
00087  mtk::DenseMatrix ff(rr,rr);
00088
00089  ff.SetValue(0,0,58.0);
00090  ff.SetValue(0,1,64.00);
00091  ff.SetValue(1,0,139.0);
00092  ff.SetValue(1,1,154.0);
00093
00094  mtk::Tools::EndUnitTestNo(1);
00095  mtk::Tools::Assert(pp == ff);
00096 }
00097
00098 int main () {
00099
00100     std::cout << "Testing mtk::BLASAdapter class." << std::endl;
00101
00102     TestRealDenseMM();
00103 }
00104
00105 #else
00106 #include <iostream>
00107 using std::cout;
00108 using std::endl;
00109 int main () {
00110     cout << "This code HAS to be compiled with support for C++11." << endl;
00111     cout << "Exiting..." << endl;
00112 }
00113 #endif

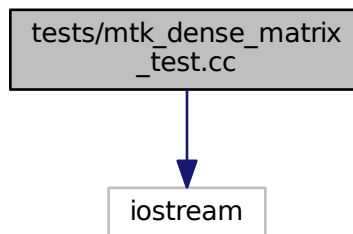
```

18.125 tests/mtk_dense_matrix_test.cc File Reference

Test file for the [mtk::DenseMatrix](#) class.

```
#include <iostream>
```

Include dependency graph for mtk_dense_matrix_test.cc:



Functions

- [int main \(\)](#)

18.125.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_dense_matrix_test.cc](#).

18.125.2 Function Documentation

18.125.2.1 int main ()

Definition at line 330 of file [mtk_dense_matrix_test.cc](#).

18.126 mtk_dense_matrix_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
```

```

00061 void TestDefaultConstructor() {
00062
00063     mtk::Tools::BeginUnitTestNo(1);
00064
00065     mtk::DenseMatrix m1;
00066
00067     mtk::Tools::EndUnitTestNo(1);
00068     mtk::Tools::Assert(m1.data() == nullptr);
00069 }
00070
00071 void TestConstructorWithNumRowsNumCols() {
00072
00073     mtk::Tools::BeginUnitTestNo(2);
00074
00075     int rr = 4;
00076     int cc = 7;
00077
00078     mtk::DenseMatrix m2(rr,cc);
00079
00080     mtk::Tools::EndUnitTestNo(2);
00081
00082     bool assertion =
00083         m2.data() != nullptr && m2.num_rows() == rr && m2.num_cols() == cc;
00084
00085     mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestConstructAsIdentity() {
00089
00090     mtk::Tools::BeginUnitTestNo(3);
00091
00092     int rank = 5;
00093     bool padded = true;
00094     bool transpose = false;
00095
00096     mtk::DenseMatrix m3(rank,padded,transpose);
00097
00098     mtk::DenseMatrix rr(rank + 2,rank);
00099
00100     for (int ii = 0; ii < rank; ++ii) {
00101         rr.SetValue(ii + 1, ii, mtk::kOne);
00102     }
00103
00104     mtk::Tools::EndUnitTestNo(3);
00105     mtk::Tools::Assert(m3 == rr);
00106 }
00107
00108 void TestConstructAsVandermonde() {
00109
00110     mtk::Tools::BeginUnitTestNo(4);
00111
00112     int rank = 5;
00113     bool padded = false;
00114     bool transpose = false;
00115
00116     mtk::DenseMatrix m4(rank,padded,transpose);
00117
00118     mtk::DenseMatrix rr(rank,rank);
00119
00120     for (int ii = 0; ii < rank; ++ii) {
00121         rr.SetValue(ii, ii, mtk::kOne);
00122     }
00123
00124     mtk::Tools::EndUnitTestNo(4);
00125     mtk::Tools::Assert(m4 == rr);
00126 }
00127
00128 void TestSetValueGetValue() {
00129
00130     mtk::Tools::BeginUnitTestNo(5);
00131
00132     int rr = 4;
00133     int cc = 7;
00134
00135     mtk::DenseMatrix m5(rr,cc);
00136
00137     for (auto ii = 0; ii < rr; ++ii) {
00138         for (auto jj = 0; jj < cc; ++jj) {
00139             m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00140         }
00141     }

```

```

00142
00143     mtk::Real *vals = m5.data();
00144
00145     bool assertion{true};
00146
00147     for (auto ii = 0; ii < rr && assertion; ++ii) {
00148         for (auto jj = 0; jj < cc && assertion; ++jj) {
00149             assertion = assertion && m5.GetValue(ii,jj) == vals[ii*cc + jj];
00150         }
00151     }
00152
00153     mtk::Tools::EndUnitTestNo(5);
00154     mtk::Tools::Assert(assertion);
00155 }
00156
00157 void TestConstructAsVandermondeTranspose() {
00158
00159     mtk::Tools::BeginUnitTestNo(6);
00160
00161     bool transpose = false;
00162     int generator_length = 3;
00163     int progression_length = 4;
00164
00165     mtk::Real generator[] = {-0.5, 0.5, 1.5};
00166
00167     mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00168
00169     transpose = true;
00170
00171     mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00172     mtk::DenseMatrix rr(progression_length, generator_length);
00173
00174     rr.SetValue(0, 0, 1.0);
00175     rr.SetValue(0, 1, 1.0);
00176     rr.SetValue(0, 2, 1.0);
00177
00178     rr.SetValue(1, 0, -0.5);
00179     rr.SetValue(1, 1, 0.5);
00180     rr.SetValue(1, 2, 1.5);
00181
00182     rr.SetValue(2, 0, 0.25);
00183     rr.SetValue(2, 1, 0.25);
00184     rr.SetValue(2, 2, 2.25);
00185
00186     rr.SetValue(3, 0, -0.125);
00187     rr.SetValue(3, 1, 0.125);
00188     rr.SetValue(3, 2, 3.375);
00189
00190     mtk::Tools::EndUnitTestNo(6);
00191     mtk::Tools::Assert(m7 == rr);
00192 }
00193
00194 void TestKron() {
00195
00196     mtk::Tools::BeginUnitTestNo(7);
00197
00198     bool padded = false;
00199     bool transpose = false;
00200     int lots_of_rows = 2;
00201     int lots_of_cols = 5;
00202     mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00203
00204     mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00205
00206     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00207         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00208             m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00209         }
00210     }
00211
00212     mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00213
00214     mtk::DenseMatrix rr(lots_of_rows*lots_of_rows, lots_of_rows*lots_of_cols);
00215
00216     rr.SetValue(0,0,1.0);
00217     rr.SetValue(0,1,2.0);
00218     rr.SetValue(0,2,3.0);
00219     rr.SetValue(0,3,4.0);
00220     rr.SetValue(0,4,5.0);
00221     rr.SetValue(0,5,0.0);
00222     rr.SetValue(0,6,0.0);

```

```

00223 rr.SetValue(0,7,0.0);
00224 rr.SetValue(0,8,0.0);
00225 rr.SetValue(0,9,0.0);
00226
00227 rr.SetValue(1,0,6.0);
00228 rr.SetValue(1,1,7.0);
00229 rr.SetValue(1,2,8.0);
00230 rr.SetValue(1,3,9.0);
00231 rr.SetValue(1,4,10.0);
00232 rr.SetValue(1,5,0.0);
00233 rr.SetValue(1,6,0.0);
00234 rr.SetValue(1,7,0.0);
00235 rr.SetValue(1,8,0.0);
00236 rr.SetValue(1,9,0.0);
00237
00238 rr.SetValue(2,0,0.0);
00239 rr.SetValue(2,1,0.0);
00240 rr.SetValue(2,2,0.0);
00241 rr.SetValue(2,3,0.0);
00242 rr.SetValue(2,4,0.0);
00243 rr.SetValue(2,5,1.0);
00244 rr.SetValue(2,6,2.0);
00245 rr.SetValue(2,7,3.0);
00246 rr.SetValue(2,8,4.0);
00247 rr.SetValue(2,9,5.0);
00248
00249 rr.SetValue(3,0,0.0);
00250 rr.SetValue(3,1,0.0);
00251 rr.SetValue(3,2,0.0);
00252 rr.SetValue(3,3,0.0);
00253 rr.SetValue(3,4,0.0);
00254 rr.SetValue(3,5,6.0);
00255 rr.SetValue(3,6,7.0);
00256 rr.SetValue(3,7,8.0);
00257 rr.SetValue(3,8,9.0);
00258 rr.SetValue(3,9,10.0);
00259
00260 mtk::Tools::EndUnitTestNo(7);
00261 mtk::Tools::Assert(m10 == rr);
00262 }
00263
00264 void TestConstructWithNumRowsNumColsAssignmentOperator() {
00265
00266     mtk::Tools::BeginUnitTestNo(8);
00267
00268     int lots_of_rows = 4;
00269     int lots_of_cols = 3;
00270     mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00271
00272     for (auto ii = 0; ii < lots_of_rows; ++ii) {
00273         for (auto jj = 0; jj < lots_of_cols; ++jj) {
00274             m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00275         }
00276     }
00277
00278     m11.Transpose();
00279
00280     mtk::DenseMatrix m12;
00281
00282     m12 = m11;
00283
00284     mtk::Tools::EndUnitTestNo(8);
00285     mtk::Tools::Assert(m11 == m12);
00286 }
00287
00288 void TestConstructAsVandermondeTransposeAssignmentOperator() {
00289
00290     mtk::Tools::BeginUnitTestNo(9);
00291
00292     bool transpose = false;
00293     int gg_l = 3;
00294     int progression_length = 4;
00295     mtk::Real gg[] = {-0.5, 0.5, 1.5};
00296
00297     mtk::DenseMatrix m13(gg, gg_l ,progression_length, transpose);
00298
00299     mtk::DenseMatrix m14;
00300
00301     m14 = m13;
00302
00303     m13.Transpose();

```

```

00304
00305     m14 = m13;
00306
00307     mtk::Tools::EndUnitTestNo(9);
00308     mtk::Tools::Assert(m13 == m14);
00309 }
00310
00311 int main () {
00312
00313     std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00314
00315     TestDefaultConstructor();
00316     TestConstructorWithNumRowsNumCols();
00317     TestConstructAsIdentity();
00318     TestConstructAsVandermonde();
00319     TestSetValueGetValue();
00320     TestConstructAsVandermondeTranspose();
00321     TestKron();
00322     TestConstructWithNumRowsNumColsAssignmentOperator();
00323     TestConstructAsVandermondeTransposeAssignmentOperator();
00324 }
00325
00326 #else
00327 #include <iostream>
00328 using std::cout;
00329 using std::endl;
00330 int main () {
00331     cout << "This code HAS to be compiled with support for C++11." << endl;
00332     cout << "Exiting..." << endl;
00333 }
00334 #endif

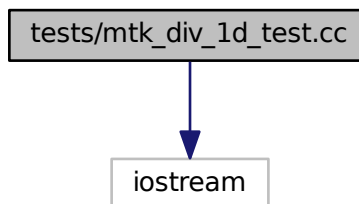
```

18.127 tests/mtk_div_1d_test.cc File Reference

Testing the mimetic 1D divergence, constructed with the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for mtk_div_1d_test.cc:



Functions

- int `main` ()

18.127.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_div_1d_test.cc](#).

18.127.2 Function Documentation**18.127.2.1 int main ()**

Definition at line 288 of file [mtk_div_1d_test.cc](#).

18.128 mtk_div_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062     mtk::Tools::BeginUnitTestNo(1);
00063 }
```



```
00064     mtk::Div1D div2;
00065
00066     bool assertion = div2.ConstructDiv1D();
00067
00068     if (!assertion) {
00069         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00070     }
00071
00072     mtk::Tools::EndUnitTestNo(1);
00073     mtk::Tools::Assert(assertion);
00074 }
00075
00076 void TestDefaultConstructorFactoryMethodFourthOrder() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Div1D div4;
00081
00082     bool assertion = div4.ConstructDiv1D(4);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00086     }
00087
00088     mtk::Tools::EndUnitTestNo(2);
00089     mtk::Tools::Assert(assertion);
00090 }
00091
00092 void TestDefaultConstructorFactoryMethodSixthOrder() {
00093
00094     mtk::Tools::BeginUnitTestNo(3);
00095
00096     mtk::Div1D div6;
00097
00098     bool assertion = div6.ConstructDiv1D(6);
00099
00100     if (!assertion) {
00101         std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00102     }
00103
00104     mtk::Tools::EndUnitTestNo(3);
00105     mtk::Tools::Assert(assertion);
00106 }
00107
00108 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00109
00110     mtk::Tools::BeginUnitTestNo(4);
00111
00112     mtk::Div1D div8;
00113
00114     bool assertion = div8.ConstructDiv1D(8);
00115
00116     if (!assertion) {
00117         std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00118     }
00119
00120     mtk::Tools::EndUnitTestNo(4);
00121     mtk::Tools::Assert(assertion);
00122 }
00123
00124 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00125
00126     mtk::Tools::BeginUnitTestNo(5);
00127
00128     mtk::Div1D div10;
00129
00130     bool assertion = div10.ConstructDiv1D(10);
00131
00132     if (!assertion) {
00133         std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00134     }
00135
00136     mtk::Tools::EndUnitTestNo(5);
00137     mtk::Tools::Assert(assertion);
00138 }
00139
00140 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00141
00142     mtk::Tools::BeginUnitTestNo(6);
00143
00144     mtk::Div1D div12;
```

```

00145
00146     bool assertion = div12.ConstructDiv1D(12);
00147
00148     if (!assertion) {
00149         std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00150     }
00151
00152     mtk::Tools::EndUnitTestNo(6);
00153     mtk::Tools::Assert(assertion);
00154 }
00155
00156 void TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold() {
00157
00158     mtk::Tools::BeginUnitTestNo(7);
00159
00160     mtk::Div1D div14;
00161
00162     bool assertion = div14.ConstructDiv1D(14);
00163
00164     if (!assertion) {
00165         std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00166     }
00167
00168     mtk::Tools::EndUnitTestNo(7);
00169     mtk::Tools::Assert(assertion);
00170 }
00171
00172 void TestSecondOrderReturnAsDenseMatrixWithGrid() {
00173
00174     mtk::Tools::BeginUnitTestNo(8);
00175
00176     mtk::Div1D div2;
00177
00178     bool assertion = div2.ConstructDiv1D();
00179
00180     if (!assertion) {
00181         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00182     }
00183
00184     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00185
00186     mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00187
00188     int rr{7};
00189     int cc{6};
00190
00191     mtk::DenseMatrix ref(rr, cc);
00192
00193     // Row 2.
00194     ref.SetValue(1,0,-5.0);
00195     ref.SetValue(1,1,5.0);
00196     ref.SetValue(1,2,0.0);
00197     ref.SetValue(1,3,0.0);
00198     ref.SetValue(1,4,0.0);
00199     ref.SetValue(1,5,0.0);
00200     ref.SetValue(1,6,0.0);
00201
00202     // Row 3.
00203     ref.SetValue(2,0,0.0);
00204     ref.SetValue(2,1,-5.0);
00205     ref.SetValue(2,2,5.0);
00206     ref.SetValue(2,3,0.0);
00207     ref.SetValue(2,4,0.0);
00208     ref.SetValue(2,5,0.0);
00209     ref.SetValue(2,6,0.0);
00210
00211     // Row 4.
00212     ref.SetValue(3,0,0.0);
00213     ref.SetValue(3,1,0.0);
00214     ref.SetValue(3,2,-5.0);
00215     ref.SetValue(3,3,5.0);
00216     ref.SetValue(3,4,0.0);
00217     ref.SetValue(3,5,0.0);
00218     ref.SetValue(3,6,0.0);
00219
00220     // Row 5.
00221     ref.SetValue(4,0,0.0);
00222     ref.SetValue(4,1,0.0);
00223     ref.SetValue(4,2,0.0);
00224     ref.SetValue(4,3,-5.0);
00225     ref.SetValue(4,4,5.0);

```

```

00226     ref.SetValue(4,5,0.0);
00227     ref.SetValue(4,6,0.0);
00228
00229     // Row 6.
00230     ref.SetValue(5,0,0.0);
00231     ref.SetValue(5,1,0.0);
00232     ref.SetValue(5,2,0.0);
00233     ref.SetValue(5,3,0.0);
00234     ref.SetValue(5,4,-5.0);
00235     ref.SetValue(5,5,5.0);
00236     ref.SetValue(5,6,0.0);
00237
00238     assertion = assertion && (div2m == ref);
00239
00240     mtk::Tools::EndUnitTestNo(8);
00241     mtk::Tools::Assert(assertion);
00242 }
00243
00244 void TestFourthOrderReturnAsDenseMatrixWithGrid() {
00245
00246     mtk::Tools::BeginUnitTestNo(9);
00247
00248     mtk::Div1D div4;
00249
00250     bool assertion = div4.ConstructDiv1D(4);
00251
00252     if (!assertion) {
00253         std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00254     }
00255
00256     std::cout << div4 << std::endl;
00257
00258     mtk::UniStgGrid1D grid(0.0, 1.0, 11);
00259
00260     std::cout << grid << std::endl;
00261
00262     mtk::DenseMatrix div4m(div4.ReturnAsDenseMatrix(grid));
00263
00264     std::cout << div4m << std::endl;
00265
00266     mtk::Tools::EndUnitTestNo(9);
00267 }
00268
00269 int main () {
00270
00271     std::cout << "Testing mtk::Div1D class." << std::endl;
00272
00273     TestDefaultConstructorFactoryMethodDefault();
00274     TestDefaultConstructorFactoryMethodFourthOrder();
00275     TestDefaultConstructorFactoryMethodSixthOrder();
00276     TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00277     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00278     TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold();
00279     TestDefaultConstructorFactoryMethodFourteenthOrderDefThreshold();
00280     TestSecondOrderReturnAsDenseMatrixWithGrid();
00281     TestFourthOrderReturnAsDenseMatrixWithGrid();
00282 }
00283
00284 #else
00285 #include <iostream>
00286 using std::cout;
00287 using std::endl;
00288 int main () {
00289     cout << "This code HAS to be compiled with support for C++11." << endl;
00290     cout << "Exiting..." << endl;
00291 }
00292 #endif

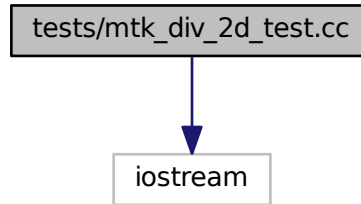
```

18.129 tests/mtk_div_2d_test.cc File Reference

Test file for the `mtk::Div2D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_div_2d_test.cc`:



Functions

- `int main ()`

18.129.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](mailto:esanchez@mail.sdsu.edu)

Definition in file [mtk_div_2d_test.cc](#).

18.129.2 Function Documentation

18.129.2.1 `int main ()`

Definition at line [139](#) of file [mtk_div_2d_test.cc](#).

18.130 `mtk_div_2d_test.cc`

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,

```

```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <cmath>
00056 #include <ctime>
00057 #include <iostream>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactory() {
00061     mtk::Tools::BeginUnitTestNo(1);
00062
00063     mtk::Div2D dd;
00064
00065     mtk::Real aa = 0.0;
00066     mtk::Real bb = 1.0;
00067     mtk::Real cc = 0.0;
00068     mtk::Real ee = 1.0;
00069
00070     int nn = 5;
00071     int mm = 5;
00072
00073     mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00074
00075     bool assertion = dd.ConstructDiv2D(ddg);
00076
00077     if (!assertion) {
00078         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00079     }
00080
00081     mtk::Tools::EndUnitTestNo(1);
00082     mtk::Tools::Assert(assertion);
00083 }
00084
00085 void TestReturnAsDenseMatrixWriteToFile() {
00086     mtk::Tools::BeginUnitTestNo(2);
00087
00088     mtk::Div2D dd;
00089
00090     mtk::Real aa = 0.0;
00091     mtk::Real bb = 1.0;
00092     mtk::Real cc = 0.0;
00093     mtk::Real ee = 1.0;
00094
00095     int nn = 5;
00096     int mm = 5;
00097
00098     mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00099
00100     bool assertion = dd.ConstructDiv2D(ddg);

```

```

00106
00107     if (!assertion) {
00108         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00109     }
00110
00111     mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00112
00113     assertion = assertion && (ddm.num_rows() != mtk::kZero);
00114
00115     std::cout << ddm << std::endl;
00116
00117     assertion = assertion && ddm.WriteToFile("mtk_div_2d_test_02.dat");
00118
00119     if (!assertion) {
00120         std::cerr << "Error writing to file." << std::endl;
00121     }
00122
00123     mtk::Tools::EndUnitTestNo(2);
00124     mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129     std::cout << "Testing mtk::Div2D class." << std::endl;
00130
00131     TestDefaultConstructorFactory();
00132     TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140     cout << "This code HAS to be compiled with support for C++11." << endl;
00141     cout << "Exiting..." << endl;
00142 }
00143 #endif

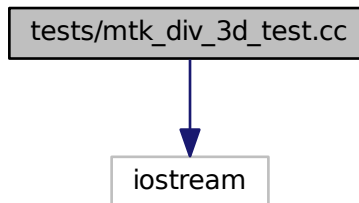
```

18.131 tests/mtk_div_3d_test.cc File Reference

Test file for the `mtk::Div3D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_div_3d_test.cc`:



Functions

- int `main()`

18.131.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_div_3d_test.cc](#).

18.131.2 Function Documentation

18.131.2.1 int main ()

Definition at line 145 of file [mtk_div_3d_test.cc](#).

18.132 mtk_div_3d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
```

```

00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Div3D div;
00068
00069     mtk::Real aa = 0.0;
00070     mtk::Real bb = 1.0;
00071     mtk::Real cc = 0.0;
00072     mtk::Real dd = 1.0;
00073     mtk::Real ee = 0.0;
00074     mtk::Real ff = 1.0;
00075
00076     int nn = 5;
00077     int mm = 5;
00078     int oo = 5;
00079
00080     mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00081
00082     bool assertion = div.ConstructDiv3D(divg);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00086     }
00087
00088     mtk::Tools::EndUnitTestNo(1);
00089     mtk::Tools::Assert(assertion);
00090 }
00091
00092 void TestReturnAsDenseMatrixWriteToFile() {
00093
00094     mtk::Tools::BeginUnitTestNo(2);
00095
00096     mtk::Div3D div;
00097
00098     mtk::Real aa = 0.0;
00099     mtk::Real bb = 1.0;
00100     mtk::Real cc = 0.0;
00101     mtk::Real dd = 1.0;
00102     mtk::Real ee = 0.0;
00103     mtk::Real ff = 1.0;
00104
00105     int nn = 5;
00106     int mm = 5;
00107     int oo = 5;
00108
00109     mtk::UniStgGrid3D divg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00110
00111     bool assertion = div.ConstructDiv3D(divg);
00112
00113     if (!assertion) {
00114         std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00115     }
00116
00117     mtk::DenseMatrix divm(div.ReturnAsDenseMatrix());
00118
00119     assertion = assertion && (divm.num_rows() != mtk::kZero);
00120
00121     std::cout << divm << std::endl;
00122
00123     assertion = assertion && divm.WriteToFile("mtk_div_3d_test_02.dat");
00124
00125     if (!assertion) {
00126         std::cerr << "Error writing to file." << std::endl;
00127     }
00128
00129     mtk::Tools::EndUnitTestNo(2);
00130     mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135     std::cout << "Testing mtk::Div3D class." << std::endl;
00136
00137     TestDefaultConstructorFactory();
00138     TestReturnAsDenseMatrixWriteToFile();
00139 }
00140
00141 #else

```



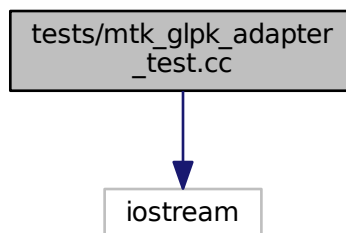
```
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146     cout << "This code HAS to be compiled with support for C++11." << endl;
00147     cout << "Exiting..." << endl;
00148 }
00149 #endif
```

18.133 tests/mtk_glpk_adapter_test.cc File Reference

Test file for the [mtk::GLPKAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for mtk_glpk_adapter_test.cc:



Functions

- `int main ()`

18.133.1 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](#).

18.133.2 Function Documentation

18.133.2.1 `int main ()`

Definition at line [81](#) of file [mtk_glpk_adapter_test.cc](#).

18.134 mtk_glpk_adapter_test.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072     std::cout << "Testing mtk::GLPKAdapter class." << std::endl;
00073
00074     Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082     cout << "This code HAS to be compiled with support for C++11." << endl;
00083     cout << "Exiting..." << endl;
00084 }
00085 #endif

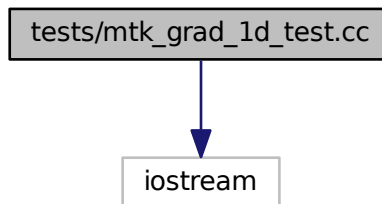
```

18.135 tests/mtk_grad_1d_test.cc File Reference

Testing the mimetic 1D gradient, constructed with the CBS algorithm.

```
#include <iostream>
```

Include dependency graph for mtk_grad_1d_test.cc:



Functions

- `int main ()`

18.135.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_1d_test.cc](#).

18.135.2 Function Documentation

18.135.2.1 `int main ()`

Definition at line [319](#) of file [mtk_grad_1d_test.cc](#).

18.136 mtk_grad_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.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.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <iostream>
00056 #include "mtk.h"
00057
00058 void TestDefaultConstructorFactoryMethodDefault() {
00059
00060     mtk::Tools::BeginUnitTestNo(1);
00061
00062     mtk::Grad1D grad2;
00063
00064     bool assertion = grad2.ConstructGrad1D();
00065
00066     if (!assertion) {
00067         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00068     }
00069
00070     std::cout << grad2 << std::endl;
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::Grad1D grad4;
00081
00082     bool assertion = grad4.ConstructGrad1D(4);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00086     }
00087
00088     std::cout << grad4 << std::endl;
00089
00090     mtk::Tools::EndUnitTestNo(2);
00091     mtk::Tools::Assert(assertion);
00092 }
00093
00094 void TestDefaultConstructorFactoryMethodSixthOrder() {
00095
00096     mtk::Tools::BeginUnitTestNo(3);

```

```

00100
00101     mtk::Grad1D grad6;
00102
00103     bool assertion = grad6.ConstructGrad1D(6);
00104
00105     if (!assertion) {
00106         std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00107     }
00108
00109     std::cout << grad6 << std::endl;
00110
00111     mtk::Tools::EndUnitTestNo(3);
00112     mtk::Tools::Assert(assertion);
00113 }
00114
00115 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00116
00117     mtk::Tools::BeginUnitTestNo(4);
00118
00119     mtk::Grad1D grad8;
00120
00121     bool assertion = grad8.ConstructGrad1D(8);
00122
00123     if (!assertion) {
00124         std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00125     }
00126
00127     std::cout << grad8 << std::endl;
00128
00129     mtk::Tools::EndUnitTestNo(4);
00130     mtk::Tools::Assert(assertion);
00131 }
00132
00133 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00134
00135     mtk::Tools::BeginUnitTestNo(5);
00136
00137     mtk::Grad1D grad10;
00138
00139     bool assertion = grad10.ConstructGrad1D(10);
00140
00141     if (!assertion) {
00142         std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00143     }
00144
00145     std::cout << grad10 << std::endl;
00146
00147     mtk::Tools::EndUnitTestNo(5);
00148     mtk::Tools::Assert(assertion);
00149 }
00150
00151 void TestReturnAsDenseMatrixWithGrid() {
00152
00153     mtk::Tools::BeginUnitTestNo(6);
00154
00155     mtk::Grad1D grad2;
00156
00157     bool assertion = grad2.ConstructGrad1D();
00158
00159     if (!assertion) {
00160         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00161     }
00162
00163     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00164
00165     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00166
00167     int rr{6};
00168     int cc{7};
00169
00170     mtk::DenseMatrix ref(rr, cc);
00171
00172     // Row 1.
00173     ref.SetValue(0,0,-13.3333);
00174     ref.SetValue(0,1,15);
00175     ref.SetValue(0,2,-1.66667);
00176     ref.SetValue(0,3,0.0);
00177     ref.SetValue(0,4,0.0);
00178     ref.SetValue(0,5,0.0);
00179     ref.SetValue(0,6,0.0);
00180

```

```

00181 // Row 2.
00182 ref.SetValue(1,0,0.0);
00183 ref.SetValue(1,1,-5.0);
00184 ref.SetValue(1,2,5.0);
00185 ref.SetValue(1,3,0.0);
00186 ref.SetValue(1,4,0.0);
00187 ref.SetValue(1,5,0.0);
00188 ref.SetValue(1,6,0.0);
00189
00190 // Row 3.
00191 ref.SetValue(2,0,0.0);
00192 ref.SetValue(2,1,0.0);
00193 ref.SetValue(2,2,-5.0);
00194 ref.SetValue(2,3,5.0);
00195 ref.SetValue(2,4,0.0);
00196 ref.SetValue(2,5,0.0);
00197 ref.SetValue(2,6,0.0);
00198
00199 // Row 4.
00200 ref.SetValue(3,0,0.0);
00201 ref.SetValue(3,1,0.0);
00202 ref.SetValue(3,2,0.0);
00203 ref.SetValue(3,3,-5.0);
00204 ref.SetValue(3,4,5.0);
00205 ref.SetValue(3,5,0.0);
00206 ref.SetValue(3,6,0.0);
00207
00208 // Row 5.
00209 ref.SetValue(4,0,0.0);
00210 ref.SetValue(4,1,0.0);
00211 ref.SetValue(4,2,0.0);
00212 ref.SetValue(4,3,0.0);
00213 ref.SetValue(4,4,-5.0);
00214 ref.SetValue(4,5,5.0);
00215 ref.SetValue(4,6,0.0);
00216
00217 // Row 6.
00218 ref.SetValue(5,0,0.0);
00219 ref.SetValue(5,1,0.0);
00220 ref.SetValue(5,2,0.0);
00221 ref.SetValue(5,3,0.0);
00222 ref.SetValue(5,4,1.66667);
00223 ref.SetValue(5,5,-15.0);
00224 ref.SetValue(5,6,13.3333);
00225
00226 mtk::Tools::EndUnitTestNo(6);
00227 mtk::Tools::Assert(grad2m == ref);
00228 }
00229
00230 void TestReturnAsDimensionlessDenseMatrix() {
00231
00232     mtk::Tools::BeginUnitTestNo(7);
00233
00234     mtk::Grad1D grad4;
00235
00236     bool assertion = grad4.ConstructGrad1D(4);
00237
00238     if (!assertion) {
00239         std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00240     }
00241
00242     mtk::DenseMatrix grad4m(grad4.ReturnAsDimensionlessDenseMatrix
(10));
00243
00244     std::cout << grad4m << std::endl;
00245
00246     mtk::Tools::EndUnitTestNo(7);
00247     mtk::Tools::Assert(assertion);
00248 }
00249
00250 void TestWriteToFile() {
00251
00252     mtk::Tools::BeginUnitTestNo(8);
00253
00254     mtk::Grad1D grad2;
00255
00256     bool assertion = grad2.ConstructGrad1D();
00257
00258     if (!assertion) {
00259         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00260     }

```

```

00261
00262     mtk::UniStgGrid1D grid(0.0, 1.0, 50);
00263
00264     mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00265
00266     std::cout << grad2m << std::endl;
00267
00268     assertion = assertion && grad2m.WriteToFile("mtk_grad_1d_test_08.dat");
00269
00270     if(!assertion) {
00271         std::cerr << "Error writing to file." << std::endl;
00272     }
00273
00274     mtk::Tools::EndUnitTestNo(8);
00275     mtk::Tools::Assert(assertion);
00276 }
00277
00278 void TestMimBndy() {
00279
00280     mtk::Tools::BeginUnitTestNo(9);
00281
00282     mtk::Grad1D grad2;
00283
00284     bool assertion = grad2.ConstructGrad1D();
00285
00286     if (!assertion) {
00287         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00288     }
00289
00290     std::cout << grad2 << std::endl;
00291
00292     mtk::DenseMatrix grad2m(grad2.mim_bndy());
00293
00294     std::cout << grad2m << std::endl;
00295
00296     mtk::Tools::EndUnitTestNo(9);
00297     mtk::Tools::Assert(assertion);
00298 }
00299
00300 int main () {
00301
00302     std::cout << "Testing mtk::Grad1D class." << std::endl;
00303
00304     TestDefaultConstructorFactoryMethodDefault();
00305     TestDefaultConstructorFactoryMethodFourthOrder();
00306     TestDefaultConstructorFactoryMethodSixthOrder();
00307     TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00308     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00309     TestReturnAsDenseMatrixWithGrid();
00310     TestReturnAsDimensionlessDenseMatrix();
00311     TestWriteToFile();
00312     TestMimBndy();
00313 }
00314
00315 #else
00316 #include <iostream>
00317 using std::cout;
00318 using std::endl;
00319 int main () {
00320     cout << "This code HAS to be compiled with support for C++11." << endl;
00321     cout << "Exiting..." << endl;
00322 }
00323 #endif

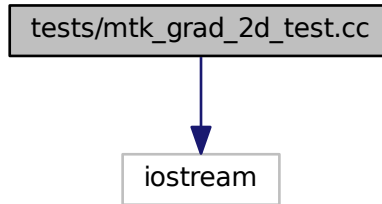
```

18.137 tests/mtk_grad_2d_test.cc File Reference

Test file for the `mtk::Grad2D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_grad_2d_test.cc`:



Functions

- `int main ()`

18.137.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_2d_test.cc](#).

18.137.2 Function Documentation

18.137.2.1 `int main ()`

Definition at line [139](#) of file [mtk_grad_2d_test.cc](#).

18.138 `mtk_grad_2d_test.cc`

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
  
```



```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Grad2D gg;
00068
00069     mtk::Real aa = 0.0;
00070     mtk::Real bb = 1.0;
00071     mtk::Real cc = 0.0;
00072     mtk::Real dd = 1.0;
00073
00074     int nn = 5;
00075     int mm = 5;
00076
00077     mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm,
00078 mtk::FieldNature::VECTOR);
00079
00080     bool assertion = gg.ConstructGrad2D(ggg);
00081
00082     if (!assertion) {
00083         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00084     }
00085     mtk::Tools::EndUnitTestNo(1);
00086     mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091     mtk::Tools::BeginUnitTestNo(2);
00092
00093     mtk::Grad2D gg;
00094
00095     mtk::Real aa = 0.0;
00096     mtk::Real bb = 1.0;
00097     mtk::Real cc = 0.0;
00098     mtk::Real dd = 1.0;
00099
00100     int nn = 5;
00101     int mm = 5;
00102
00103     mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm,
00104 mtk::FieldNature::VECTOR);

```

```

00104
00105     bool assertion = gg.ConstructGrad2D(ggg);
00106
00107     if (!assertion) {
00108         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00109     }
00110
00111     mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00112
00113     assertion = assertion && (ggm.num_rows() != mtk::kZero);
00114
00115     std::cout << ggm << std::endl;
00116
00117     assertion = assertion && ggm.WriteToFile("mtk_grad_2d_test_02.dat");
00118
00119     if(!assertion) {
00120         std::cerr << "Error writing to file." << std::endl;
00121     }
00122
00123     mtk::Tools::EndUnitTestNo(2);
00124     mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128     std::cout << "Testing mtk::Grad2D class." << std::endl;
00129
00130     TestDefaultConstructorFactory();
00131     TestReturnAsDenseMatrixWriteToFile();
00132 }
00133
00134 #else
00135 #include <iostream>
00136 using std::cout;
00137 using std::endl;
00138
00139 int main () {
00140     cout << "This code HAS to be compiled with support for C++11." << endl;
00141     cout << "Exiting..." << endl;
00142 }
00143 #endif

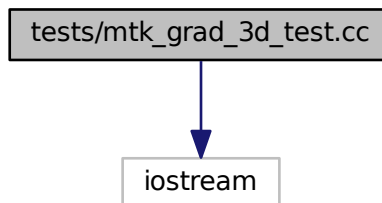
```

18.139 tests/mtk_grad_3d_test.cc File Reference

Test file for the [mtk::Grad3D](#) class.

```
#include <iostream>
```

Include dependency graph for mtk_grad_3d_test.cc:



Functions

- `int main ()`

18.139.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_grad_3d_test.cc](#).

18.139.2 Function Documentation

18.139.2.1 `int main ()`

Definition at line 147 of file [mtk_grad_3d_test.cc](#).

18.140 mtk_grad_3d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
```

```

00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Grad3D gg;
00068
00069     mtk::Real aa = 0.0;
00070     mtk::Real bb = 1.0;
00071     mtk::Real cc = 0.0;
00072     mtk::Real dd = 1.0;
00073     mtk::Real ee = 0.0;
00074     mtk::Real ff = 1.0;
00075
00076     int nn = 5;
00077     int mm = 5;
00078     int oo = 5;
00079
00080     mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo,
00081 mtk::FieldNature::VECTOR);
00082
00083     bool assertion = gg.ConstructGrad3D(ggg);
00084
00085     if (!assertion) {
00086         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00087     }
00088
00089     mtk::Tools::EndUnitTestNo(1);
00090     mtk::Tools::Assert(assertion);
00091 }
00092
00093 void TestReturnAsDenseMatrixWriteToFile() {
00094
00095     mtk::Tools::BeginUnitTestNo(2);
00096
00097     mtk::Grad3D gg;
00098
00099     mtk::Real aa = 0.0;
00100     mtk::Real bb = 1.0;
00101     mtk::Real cc = 0.0;
00102     mtk::Real dd = 1.0;
00103     mtk::Real ee = 0.0;
00104     mtk::Real ff = 1.0;
00105
00106     int nn = 5;
00107     int mm = 5;
00108     int oo = 5;
00109
00110     mtk::UniStgGrid3D ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo,
00111 mtk::FieldNature::VECTOR);
00112
00113     bool assertion = gg.ConstructGrad3D(ggg);
00114
00115     if (!assertion) {
00116         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00117     }
00118
00119     mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00120
00121     assertion = assertion && (ggm.num_rows() != mtk::kZero);
00122
00123     std::cout << ggm << std::endl;
00124
00125     assertion = assertion && ggm.WriteToFile("mtk_grad_3d_test_02.dat");
00126
00127     if (!assertion) {
00128         std::cerr << "Error writing to file." << std::endl;
00129     }
00130
00131     mtk::Tools::EndUnitTestNo(2);
00132     mtk::Tools::Assert(assertion);
00133 }
00134

```

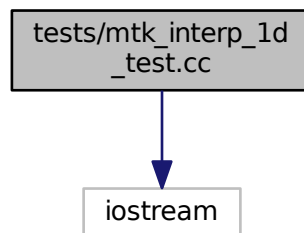
```
00135 int main () {
00136
00137     std::cout << "Testing mtk::Grad2D class." << std::endl;
00138
00139     TestDefaultConstructorFactory();
00140     TestReturnAsDenseMatrixWriteToFile();
00141 }
00142
00143 #else
00144 #include <iostream>
00145 using std::cout;
00146 using std::endl;
00147 int main () {
00148     cout << "This code HAS to be compiled with support for C++11." << endl;
00149     cout << "Exiting..." << endl;
00150 }
00151 #endif
```

18.141 tests/mtk_interp_1d_test.cc File Reference

Testing the 1D interpolation.

```
#include <iostream>
```

Include dependency graph for mtk_interp_1d_test.cc:



Functions

- int `main` ()

18.141.1 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](#).

18.141.2 Function Documentation

18.141.2.1 int main ()

Definition at line 113 of file [mtk_interp_1d_test.cc](#).

18.142 mtk_interp_1d_test.cc

```

00001  /*
00010  /*
00011  Copyright (C) 2015, Computational Science Research Center, San Diego State
00012  University. All rights reserved.
00013
00014  Redistribution and use in source and binary forms, with or without modification,
00015  are permitted provided that the following conditions are met:
00016
00017  1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018  and a copy of the modified files should be reported once modifications are
00019  completed, unless these modifications are made through the project's GitHub
00020  page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021  should be developed and included in any deliverable.
00022
00023  2. Redistributions of source code must be done through direct
00024  downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026  3. Redistributions in binary form must reproduce the above copyright notice,
00027  this list of conditions and the following disclaimer in the documentation and/or
00028  other materials provided with the distribution.
00029
00030  4. Usage of the binary form on proprietary applications shall require explicit
00031  prior written permission from the the copyright holders, and due credit should
00032  be given to the copyright holders.
00033
00034  5. Neither the name of the copyright holder nor the names of its contributors
00035  may be used to endorse or promote products derived from this software without
00036  specific prior written permission.
00037
00038  The copyright holders provide no reassurances that the source code provided does
00039  not infringe any patent, copyright, or any other intellectual property rights of
00040  third parties. The copyright holders disclaim any liability to any recipient for
00041  claims brought against recipient by any third party for infringement of that
00042  parties intellectual property rights.
00043
00044  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045  ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048  ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051  ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054  */
00055
00056  #if __cplusplus == 201103L
00057
00058  #include <iostream>
00059
00060  #include "mtk.h"
00061
00062  void TestDefaultConstructorFactoryMethodDefault() {
00063
00064      mtk::Tools::BeginUnitTestNo(1);
00065
00066      mtk::Interp1D inter;
00067
00068      bool assertion = inter.ConstructInterp1D();
00069
00070      if (!assertion) {
00071          std::cerr << "Mimetic interp could not be built." << std::endl;
00072      }
00073
00074      mtk::Tools::EndUnitTestNo(1);
00075      mtk::Tools::Assert(assertion);
00076  }
00077
00078  void TestReturnAsDenseMatrixWithGrid() {

```

```

00079
00080     mtk::Tools::BeginUnitTestNo(2);
00081
00082     mtk::Interp1D inter;
00083
00084     bool assertion = inter.ConstructInterp1D();
00085
00086     if (!assertion) {
00087         std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00088     }
00089
00090     mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00091
00092     mtk::DenseMatrix interpm(inter.ReturnAsDenseMatrix(grid));
00093
00094     assertion =
00095         assertion && interpm.GetValue(0,0) == 1.0 && interpm.GetValue(5,6) == 1.0;
00096
00097     mtk::Tools::EndUnitTestNo(2);
00098     mtk::Tools::Assert(assertion);
00099 }
00100
00101 int main () {
00102
00103     std::cout << "Testing mtk::Interp1D class." << std::endl;
00104
00105     TestDefaultConstructorFactoryMethodDefault();
00106     TestReturnAsDenseMatrixWithGrid();
00107 }
00108
00109 #else
00110 #include <iostream>
00111 using std::cout;
00112 using std::endl;
00113 int main () {
00114     cout << "This code HAS to be compiled with support for C++11." << endl;
00115     cout << "Exiting..." << endl;
00116 }
00117 #endif

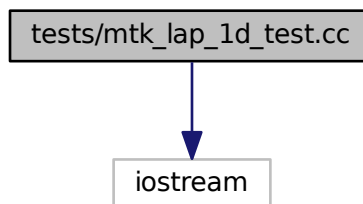
```

18.143 tests/mtk_lap_1d_test.cc File Reference

Testing the 1D Laplacian operator.

```
#include <iostream>
```

Include dependency graph for mtk_lap_1d_test.cc:



Functions

- int [main](#) ()

18.143.1 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](#).

18.143.2 Function Documentation

18.143.2.1 int main ()

Definition at line 193 of file [mtk_lap_1d_test.cc](#).

18.144 mtk_lap_1d_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059
00060 #include "mtk.h"
```



```
00061
00062 void TestDefaultConstructorFactoryMethodDefault() {
00063
00064     mtk::Tools::BeginUnitTestNo(1);
00065
00066     mtk::Lap1D lap2;
00067
00068     bool assertion = lap2.ConstructLap1D();
00069
00070     if (!assertion) {
00071         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00072     }
00073
00074     mtk::Tools::EndUnitTestNo(1);
00075     mtk::Tools::Assert(assertion);
00076 }
00077
00078 void TestDefaultConstructorFactoryMethodFourthOrder() {
00079
00080     mtk::Tools::BeginUnitTestNo(2);
00081
00082     mtk::Lap1D lap4;
00083
00084     bool assertion = lap4.ConstructLap1D(4);
00085
00086     if (!assertion) {
00087         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00088     }
00089
00090     mtk::Tools::EndUnitTestNo(2);
00091     mtk::Tools::Assert(assertion);
00092 }
00093
00094 void TestDefaultConstructorFactoryMethodSixthOrder() {
00095
00096     mtk::Tools::BeginUnitTestNo(3);
00097
00098     mtk::Lap1D lap6;
00099
00100     bool assertion = lap6.ConstructLap1D(6);
00101
00102     if (!assertion) {
00103         std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00104     }
00105
00106     mtk::Tools::EndUnitTestNo(3);
00107     mtk::Tools::Assert(assertion);
00108 }
00109
00110 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00111
00112     mtk::Tools::BeginUnitTestNo(4);
00113
00114     mtk::Lap1D lap8;
00115
00116     bool assertion = lap8.ConstructLap1D(8);
00117
00118     if (!assertion) {
00119         std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00120     }
00121
00122     mtk::Tools::EndUnitTestNo(4);
00123 }
00124
00125 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00126
00127     mtk::Tools::BeginUnitTestNo(5);
00128
00129     mtk::Lap1D lap10;
00130
00131     bool assertion = lap10.ConstructLap1D(10);
00132
00133     if (!assertion) {
00134         std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00135     }
00136
00137     mtk::Tools::EndUnitTestNo(5);
00138 }
00139
00140 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00141
```

```

00142     mtk::Tools::BeginUnitTestNo(6);
00143
00144     mtk::Lap1D lap12;
00145
00146     bool assertion = lap12.ConstructLap1D(12);
00147
00148     if (!assertion) {
00149         std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00150     }
00151
00152     mtk::Tools::EndUnitTestNo(6);
00153 }
00154
00155 void TestReturnAsDenseMatrix() {
00156
00157     mtk::Tools::BeginUnitTestNo(8);
00158
00159     mtk::Lap1D lap4;
00160
00161     bool assertion = lap4.ConstructLap1D(4);
00162
00163     if (!assertion) {
00164         std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00165     }
00166
00167     mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00168
00169     mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00170
00171     assertion = assertion &&
00172         abs(lap4_m.GetValue(1, 0) - 385.133) < mtk::kDefaultTolerance &&
00173         abs(lap4_m.GetValue(11, 12) - 385.133) < mtk::kDefaultTolerance;
00174     mtk::Tools::EndUnitTestNo(8);
00175     mtk::Tools::Assert(assertion);
00176 }
00177
00178 int main () {
00179
00180     std::cout << "Testing MTK 1D Laplacian" << std::endl;
00181
00182     TestDefaultConstructorFactoryMethodDefault();
00183     TestDefaultConstructorFactoryMethodFourthOrder();
00184     TestDefaultConstructorFactoryMethodSixthOrder();
00185     TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00186     TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00187     TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold();
00188     TestReturnAsDenseMatrix();
00189 }
00190
00191 #else
00192 #include <iostream>
00193 int main () {
00194     std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00195     std::cout << "Exiting..." << std::endl;
00196 }
00197 #endif

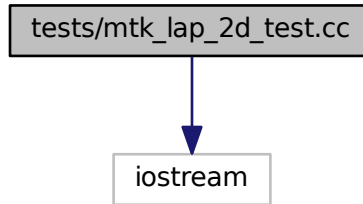
```

18.145 tests/mtk_lap_2d_test.cc File Reference

Test file for the [mtk::Lap2D](#) class.

```
#include <iostream>
```

Include dependency graph for mtk_lap_2d_test.cc:



Functions

- int [main](#) ()

18.145.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lap_2d_test.cc](#).

18.145.2 Function Documentation

18.145.2.1 int main ()

Definition at line [139](#) of file [mtk_lap_2d_test.cc](#).

18.146 mtk_lap_2d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,

```

```

00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055 #include <cmath>
00056 #include <ctime>
00057 #include <iostream>
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactory() {
00061     mtk::Tools::BeginUnitTestNo(1);
00062
00063     mtk::Lap2D ll;
00064
00065     mtk::Real aa = 0.0;
00066     mtk::Real bb = 1.0;
00067     mtk::Real cc = 0.0;
00068     mtk::Real dd = 1.0;
00069
00070     int nn = 5;
00071     int mm = 5;
00072
00073     mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00074
00075     bool assertion = ll.ConstructLap2D(llg);
00076
00077     if (!assertion) {
00078         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00079     }
00080
00081     mtk::Tools::EndUnitTestNo(1);
00082     mtk::Tools::Assert(assertion);
00083 }
00084
00085 void TestReturnAsDenseMatrixWriteToFile() {
00086     mtk::Tools::BeginUnitTestNo(2);
00087
00088     mtk::Lap2D ll;
00089
00090     mtk::Real aa = 0.0;
00091     mtk::Real bb = 1.0;
00092     mtk::Real cc = 0.0;
00093     mtk::Real dd = 1.0;
00094
00095     int nn = 5;
00096     int mm = 5;
00097
00098     mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00099
00100     bool assertion = ll.ConstructLap2D(llg);

```

```

00106
00107     if (!assertion) {
00108         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00109     }
00110
00111     mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00112
00113     assertion = assertion && (llm.num_rows() != 0);
00114
00115     std::cout << llm << std::endl;
00116
00117     assertion = assertion && llm.WriteToFile("mtk_lap_2d_test_02.dat");
00118
00119     if (!assertion) {
00120         std::cerr << "Error writing to file." << std::endl;
00121     }
00122
00123     mtk::Tools::EndUnitTestNo(2);
00124     mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129     std::cout << "Testing mtk::Lap2D class." << std::endl;
00130
00131     TestDefaultConstructorFactory();
00132     TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140     cout << "This code HAS to be compiled with support for C++11." << endl;
00141     cout << "Exiting..." << endl;
00142 }
00143 #endif

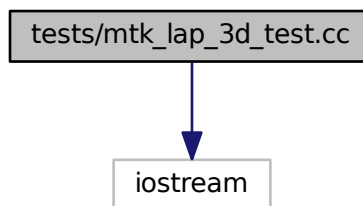
```

18.147 tests/mtk_lap_3d_test.cc File Reference

Test file for the `mtk::Lap3D` class.

```
#include <iostream>
```

Include dependency graph for `mtk_lap_3d_test.cc`:



Functions

- int `main` ()

18.147.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_lap_3d_test.cc](#).

18.147.2 Function Documentation

18.147.2.1 int main ()

Definition at line 145 of file [mtk_lap_3d_test.cc](#).

18.148 mtk_lap_3d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
```

```

00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Lap3D ll;
00068
00069     mtk::Real aa = 0.0;
00070     mtk::Real bb = 1.0;
00071     mtk::Real cc = 0.0;
00072     mtk::Real dd = 1.0;
00073     mtk::Real ee = 0.0;
00074     mtk::Real ff = 1.0;
00075
00076     int nn = 5;
00077     int mm = 5;
00078     int oo = 5;
00079
00080     mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00081
00082     bool assertion = ll.ConstructLap3D(llg);
00083
00084     if (!assertion) {
00085         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00086     }
00087
00088     mtk::Tools::EndUnitTestNo(1);
00089     mtk::Tools::Assert(assertion);
00090 }
00091
00092 void TestReturnAsDenseMatrixWriteToFile() {
00093
00094     mtk::Tools::BeginUnitTestNo(2);
00095
00096     mtk::Lap3D ll;
00097
00098     mtk::Real aa = 0.0;
00099     mtk::Real bb = 1.0;
00100     mtk::Real cc = 0.0;
00101     mtk::Real dd = 1.0;
00102     mtk::Real ee = 0.0;
00103     mtk::Real ff = 1.0;
00104
00105     int nn = 5;
00106     int mm = 5;
00107     int oo = 5;
00108
00109     mtk::UniStgGrid3D llg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00110
00111     bool assertion = ll.ConstructLap3D(llg);
00112
00113     if (!assertion) {
00114         std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00115     }
00116
00117     mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00118
00119     assertion = assertion && (llm.num_rows() != 0);
00120
00121     std::cout << llm << std::endl;
00122
00123     assertion = assertion && llm.WriteToFile("mtk_lap_3d_test_02.dat");
00124
00125     if (!assertion) {
00126         std::cerr << "Error writing to file." << std::endl;
00127     }
00128
00129     mtk::Tools::EndUnitTestNo(2);
00130     mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135     std::cout << "Testing mtk::Lap3D class." << std::endl;
00136
00137     TestDefaultConstructorFactory();
00138     TestReturnAsDenseMatrixWriteToFile();
00139 }
00140
00141 #else

```

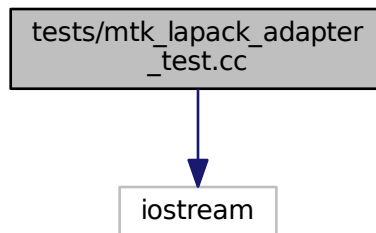
```
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146     cout << "This code HAS to be compiled with support for C++11." << endl;
00147     cout << "Exiting..." << endl;
00148 }
00149 #endif
```

18.149 tests/mtk_lapack_adapter_test.cc File Reference

Test file for the [mtk::LAPACKAdapter](#) class.

```
#include <iostream>
```

Include dependency graph for mtk_lapack_adapter_test.cc:



Functions

- `int main ()`

18.149.1 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](#).

18.149.2 Function Documentation

18.149.2.1 `int main ()`

Definition at line 81 of file [mtk_lapack_adapter_test.cc](#).

18.150 mtk_lapack_adapter_test.cc

```

00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072     std::cout << "Testing mtk::LAPACKAdapter class." << std::endl;
00073
00074     Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082     cout << "This code HAS to be compiled with support for C++11." << endl;
00083     cout << "Exiting..." << endl;
00084 }
00085 #endif

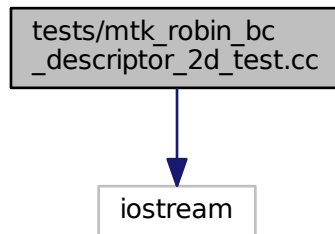
```

18.151 tests/mtk_robin_bc_descriptor_2d_test.cc File Reference

Test file for the [mtk::RobinBCDescriptor2D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_robin_bc_descriptor_2d_test.cc`:



Functions

- `int main ()`

18.151.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - [esanchez at mail dot sdsu dot edu](#)

Definition in file [mtk_robin_bc_descriptor_2d_test.cc](#).

18.151.2 Function Documentation

18.151.2.1 `int main ()`

Definition at line [198](#) of file [mtk_robin_bc_descriptor_2d_test.cc](#).

18.152 mtk_robin_bc_descriptor_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
```

```

00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorGetters() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::RobinBCDescriptor2D bcd;
00068
00069     bool assertion{true};
00070
00071     assertion = assertion && bcd.highest_order_diff_west() == -1;
00072     assertion = assertion && bcd.highest_order_diff_east() == -1;
00073     assertion = assertion && bcd.highest_order_diff_south() == -1;
00074     assertion = assertion && bcd.highest_order_diff_north() == -1;
00075
00076     mtk::Tools::EndUnitTestNo(1);
00077     mtk::Tools::Assert(assertion);
00078 }
00079
00080 mtk::Real cc(const mtk::Real &xx, const mtk::Real &yy) {
00081
00082     return mtk::kOne;
00083 }
00084
00085 void TestPushBackImposeOnLaplacianMatrix() {
00086
00087     mtk::Tools::BeginUnitTestNo(2);
00088
00089     mtk::RobinBCDescriptor2D bcd;
00090
00091     bool assertion{true};
00092
00093     bcd.PushBackWestCoeff(cc);
00094     bcd.PushBackEastCoeff(cc);
00095     bcd.PushBackSouthCoeff(cc);
00096     bcd.PushBackNorthCoeff(cc);
00097
00098     assertion = assertion && bcd.highest_order_diff_west() == 0;

```

```

00099  assertion = assertion && bcd.highest_order_diff_east() == 0;
00100  assertion = assertion && bcd.highest_order_diff_south() == 0;
00101  assertion = assertion && bcd.highest_order_diff_north() == 0;
00102
00103  mtk::Real aa = 0.0;
00104  mtk::Real bb = 1.0;
00105  mtk::Real cc = 0.0;
00106  mtk::Real dd = 1.0;
00107
00108  int nn = 5;
00109  int mm = 5;
00110
00111  mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00112
00113  mtk::Lap2D ll;
00114
00115  assertion = ll.ConstructLap2D(llg);
00116
00117  if (!assertion) {
00118      std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00119  }
00120
00121  mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00122
00123  assertion = assertion && (llm.num_rows() != 0);
00124
00125  bcd.ImposeOnLaplacianMatrix(ll, llg, llm);
00126
00127  assertion = assertion &&
00128      llm.WriteToFile("mtk_robin_bc_descriptor_2d_test_02.dat");
00129
00130  mtk::Tools::EndUnitTestNo(2);
00131  mtk::Tools::Assert(assertion);
00132 }
00133
00134 mtk::Real ScalarField(const mtk::Real &xx, const mtk::Real &yy) {
00135
00136     mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
00137
00138     return xx*yy*exp(aux);
00139 }
00140
00141 mtk::Real HomogeneousDiricheletBC(const mtk::Real &xx,
00142                                   const mtk::Real &tt) {
00143
00144     return mtk::kZero;
00145 }
00146
00147 void TestImposeOnGrid() {
00148
00149     mtk::Tools::BeginUnitTestNo(3);
00150
00151     mtk::Real aa = 0.0;
00152     mtk::Real bb = 1.0;
00153     mtk::Real cc = 0.0;
00154     mtk::Real dd = 1.0;
00155
00156     int nn = 5;
00157     int mm = 5;
00158
00159     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00160
00161     gg.BindScalarField(ScalarField);
00162
00163     mtk::RobinBCDescriptor2D desc;
00164
00165     desc.set_west_condition(HomogeneousDiricheletBC);
00166     desc.set_east_condition(HomogeneousDiricheletBC);
00167     desc.set_south_condition(HomogeneousDiricheletBC);
00168     desc.set_north_condition(HomogeneousDiricheletBC);
00169
00170     desc.ImposeOnGrid(gg);
00171
00172     bool assertion{gg.WriteToFile("mtk_robin_bc_descriptor_2d_test_03.dat",
00173                                   "x",
00174                                   "y",
00175                                   "u(x,y)");};
00176
00177     if(!assertion) {
00178         std::cerr << "Error writing to file." << std::endl;
00179     }

```

```

00180
00181     mtk::Tools::EndUnitTestNo(3);
00182     mtk::Tools::Assert(assertion);
00183 }
00184
00185 int main () {
00186     std::cout << "Testing mtk::RobinBCDescriptor2D class." << std::endl;
00187
00188     TestDefaultConstructorGetters();
00189     TestPushBackImposeOnLaplacianMatrix();
00190     TestImposeOnGrid();
00191 }
00192
00193 #else
00194 #include <iostream>
00195 using std::cout;
00196 using std::endl;
00197
00198 int main () {
00199     cout << "This code HAS to be compiled with support for C++11." << endl;
00200     cout << "Exiting..." << endl;
00201 }
00202 #endif

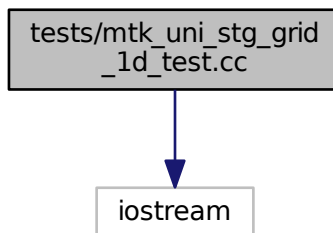
```

18.153 tests/mtk_uni_stg_grid_1d_test.cc File Reference

Test file for the [mtk::UniStgGrid1D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_uni_stg_grid_1d_test.cc`:



Functions

- `int main ()`

18.153.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_1d_test.cc](#).

18.153.2 Function Documentation

18.153.2.1 `int main ()`

Definition at line 172 of file [mtk_uni_stg_grid_1d_test.cc](#).

18.154 `mtk_uni_stg_grid_1d_test.cc`

```

00001
00002 /*
00003 Copyright (C) 2015, 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 The copyright holders provide no reassurances that the source code provided does
00031 not infringe any patent, copyright, or any other intellectual property rights of
00032 third parties. The copyright holders disclaim any liability to any recipient for
00033 claims brought against recipient by any third party for infringement of that
00034 parties intellectual property rights.
00035
00036 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00037 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00038 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00039 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00040 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00041 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00042 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00043 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00044 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00045 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00046 */
00047 #if __cplusplus == 201103L
00048
00049 #include <iostream>
00050 #include <ctime>
00051
00052 #include "mtk.h"
00053
00054 void TestDefaultConstructor() {
00055
00056     mtk::Tools::BeginUnitTestNo(1);
00057
00058     mtk::UniStgGrid1D gg;
00059
00060     mtk::Tools::EndUnitTestNo(1);
00061     mtk::Tools::Assert(gg.delta_x() == mtk::kZero);
00062 }
00063
00064 mtk::Real ScalarField(const mtk::Real &xx) {
00065
00066     return 2.0*xx;
00067
00068 }

```

```

00074 }
00075
00076 void TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Real aa = 0.0;
00081     mtk::Real bb = 1.0;
00082
00083     int nn = 5;
00084
00085     mtk::UniStgGrid1D gg(aa, bb, nn);
00086
00087     gg.BindScalarField(ScalarField);
00088
00089     std::cout << gg << std::endl;
00090
00091     mtk::Tools::EndUnitTestNo(2);
00092     mtk::Tools::Assert(gg.delta_x() == 0.2 && gg.
num_cells_x() == 5);
00093 }
00094
00095 void TestBindScalarFieldWriteToFile() {
00096
00097     mtk::Tools::BeginUnitTestNo(3);
00098
00099     mtk::Real aa = 0.0;
00100     mtk::Real bb = 1.0;
00101
00102     int nn = 5;
00103
00104     mtk::UniStgGrid1D gg(aa, bb, nn);
00105
00106     bool assertion{true};
00107
00108     gg.BindScalarField(ScalarField);
00109
00110     assertion =
00111         assertion &&
00112         gg.discrete_field()[0] == 0.0 &&
00113         gg.discrete_field()[gg.num_cells_x() + 2 - 1] == 2.0;
00114
00115     if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_03.dat", "x", "u(x)")) {
00116         std::cerr << "Error writing to file." << std::endl;
00117         assertion = false;
00118     }
00119
00120     mtk::Tools::EndUnitTestNo(3);
00121     mtk::Tools::Assert(assertion);
00122 }
00123
00124 mtk::Real VectorFieldPComponent(mtk::Real xx) {
00125
00126     return xx*xx;
00127 }
00128
00129 void TestBindVectorField() {
00130
00131     mtk::Tools::BeginUnitTestNo(4);
00132
00133     mtk::Real aa = 0.0;
00134     mtk::Real bb = 1.0;
00135
00136     int nn = 20;
00137
00138     mtk::UniStgGrid1D gg(aa, bb, nn, mtk::FieldNature::VECTOR);
00139
00140     bool assertion{true};
00141
00142     gg.BindVectorField(VectorFieldPComponent);
00143
00144     assertion =
00145         assertion &&
00146         gg.discrete_field()[0] == 0.0 &&
00147         gg.discrete_field()[gg.num_cells_x() + 1 - 1] == 1.0;
00148
00149     if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_04.dat", "x", "v(x)")) {
00150         std::cerr << "Error writing to file." << std::endl;
00151         assertion = false;
00152     }
00153 }

```

```

00154     mtk::Tools::EndUnitTestNo(4);
00155     mtk::Tools::Assert(assertion);
00156 }
00157
00158 int main () {
00159
00160     std::cout << "Testing mtk::UniStgGrid1D class." << std::endl;
00161
00162     TestDefaultConstructor();
00163     TestConstructWithWestBdyEastBdyNumCellsOStreamOperatorBindScalarField();
00164     TestBindScalarFieldWriteToFile();
00165     TestBindVectorField();
00166 }
00167
00168 #else
00169 #include <iostream>
00170 using std::cout;
00171 using std::endl;
00172 int main () {
00173     cout << "This code HAS to be compiled with support for C++11." << endl;
00174     cout << "Exiting..." << endl;
00175 }
00176 #endif

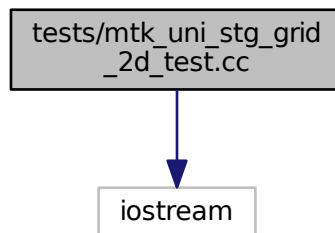
```

18.155 tests/mtk_uni_stg_grid_2d_test.cc File Reference

Test file for the [mtk::UniStgGrid2D](#) class.

```
#include <iostream>
```

Include dependency graph for `mtk_uni_stg_grid_2d_test.cc`:



Functions

- int [main](#) ()

18.155.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_2d_test.cc](#).

18.155.2 Function Documentation

18.155.2.1 int main ()

Definition at line 202 of file [mtk_uni_stg_grid_2d_test.cc](#).

18.156 mtk_uni_stg_grid_2d_test.cc

```

00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructor() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::UniStgGrid2D gg;
00068
00069     mtk::Tools::EndUnitTestNo(1);
00070     mtk::Tools::Assert(gg.delta_x() == mtk::kZero && gg.
00071     delta_y() == mtk::kZero);
00072 }
00073

```

```

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
00083     int nn = 5;
00084     int mm = 7;
00085
00086     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00087
00088     std::cout << gg << std::endl;
00089
00090     mtk::Tools::EndUnitTestNo(2);
00091     mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00092                       abs(gg.delta_y() - 0.142857) <
00093                       mtk::kDefaultTolerance);
00094 }
00095 void TestGetters() {
00096
00097     mtk::Tools::BeginUnitTestNo(3);
00098
00099     mtk::Real aa = 0.0;
00100     mtk::Real bb = 1.0;
00101     mtk::Real cc = 0.0;
00102     mtk::Real dd = 1.0;
00103
00104     int nn = 5;
00105     int mm = 7;
00106
00107     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00108
00109     bool assertion{true};
00110
00111     assertion = assertion && (gg.west_bndy() == aa);
00112     assertion = assertion && (gg.east_bndy() == bb);
00113     assertion = assertion && (gg.num_cells_x() == nn);
00114     assertion = assertion && (gg.south_bndy() == cc);
00115     assertion = assertion && (gg.north_bndy() == dd);
00116     assertion = assertion && (gg.num_cells_y() == mm);
00117
00118     mtk::Tools::EndUnitTestNo(3);
00119     mtk::Tools::Assert(assertion);
00120 }
00121
00122 mtk::Real ScalarField(const mtk::Real &xx, const mtk::Real &yy) {
00123
00124     mtk::Real aux{-(1.0/2.0)*xx*xx - (1.0/2.0)*yy*yy};
00125
00126     return xx*yy*exp(aux);
00127 }
00128
00129 void TestBindScalarFieldWriteToFile() {
00130
00131     mtk::Tools::BeginUnitTestNo(4);
00132
00133     mtk::Real aa = 0.0;
00134     mtk::Real bb = 1.0;
00135     mtk::Real cc = 0.0;
00136     mtk::Real dd = 1.0;
00137
00138     int nn = 5;
00139     int mm = 5;
00140
00141     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm);
00142
00143     gg.BindScalarField(ScalarField);
00144
00145     if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_04.dat", "x", "y", "u(x,y)")) {
00146         std::cerr << "Error writing to file." << std::endl;
00147     }
00148
00149     mtk::Tools::EndUnitTestNo(4);
00150 }
00151
00152 mtk::Real VectorFieldPComponent(const mtk::Real &xx, const

```

```

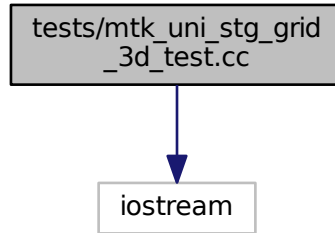
    mtk::Real &yy) {
00153
00154     return xx + 0.01;
00155 }
00156
00157 mtk::Real VectorFieldQComponent(const mtk::Real &xx, const
    mtk::Real &yy) {
00158
00159     return yy + 0.01;
00160 }
00161
00162 void TestBindVectorField() {
00163
00164     mtk::Tools::BeginUnitTestNo(5);
00165
00166     mtk::Real aa = 0.0;
00167     mtk::Real bb = 1.0;
00168     mtk::Real cc = 0.0;
00169     mtk::Real dd = 1.0;
00170
00171     int nn = 5;
00172     int mm = 5;
00173
00174     mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm,
    mtk::FieldNature::VECTOR);
00175
00176     gg.BindVectorField(VectorFieldPComponent, VectorFieldQComponent);
00177
00178     std::cout << gg << std::endl;
00179
00180     if(!gg.WriteToFile("mtk_uni_stg_grid_2d_test_05.dat", "x", "y", "v(x,y)")) {
00181         std::cerr << "Error writing to file." << std::endl;
00182     }
00183
00184     mtk::Tools::EndUnitTestNo(5);
00185 }
00186
00187 int main () {
00188
00189     std::cout << "Testing mtk::UniStgGrid2D class." << std::endl;
00190
00191     TestDefaultConstructor();
00192     TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYostreamOperator();
00193     TestGetters();
00194     TestBindScalarFieldWriteToFile();
00195     TestBindVectorField();
00196 }
00197
00198 #else
00199 #include <iostream>
00200 using std::cout;
00201 using std::endl;
00202 int main () {
00203     cout << "This code HAS to be compiled with support for C++11." << endl;
00204     cout << "Exiting..." << endl;
00205 }
00206 #endif

```

18.157 tests/mtk_uni_stg_grid_3d_test.cc File Reference

Test file for the [mtk::UniStgGrid3D](#) class.

```
#include <iostream>
Include dependency graph for mtk_uni_stg_grid_3d_test.cc:
```



Functions

- int [main](#) ()

18.157.1 Detailed Description

Author

: Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu

Definition in file [mtk_uni_stg_grid_3d_test.cc](#).

18.157.2 Function Documentation

18.157.2.1 int main ()

Definition at line [184](#) of file [mtk_uni_stg_grid_3d_test.cc](#).

18.158 mtk_uni_stg_grid_3d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
```

```

00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructor() {
00064
00065     mtk::Tools::BeginUnitTestNo(1);
00066
00067     mtk::UniStgGrid3D gg;
00068
00069     mtk::Tools::EndUnitTestNo(1);
00070     mtk::Tools::Assert(gg.delta_x() == mtk::kZero &&
00071                       gg.delta_y() == mtk::kZero &&
00072                       gg.delta_z() == mtk::kZero);
00073 }
00074
00075 void
00076 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYostreamOperator() {
00077
00078     mtk::Tools::BeginUnitTestNo(2);
00079
00080     mtk::Real aa = 0.0;
00081     mtk::Real bb = 1.0;
00082     mtk::Real cc = 0.0;
00083     mtk::Real dd = 1.0;
00084     mtk::Real ee = 0.0;
00085     mtk::Real ff = 1.0;
00086
00087     int nn = 5;
00088     int mm = 7;
00089     int oo = 7;
00090
00091     mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00092
00093     std::cout << gg << std::endl;
00094
00095     mtk::Tools::EndUnitTestNo(2);
00096     mtk::Tools::Assert(gg.delta_x() == 0.2 &&
00097                       abs(gg.delta_y() - 0.142857) <
00098                       mtk::kDefaultTolerance);
00099 }
00100 void TestGetters() {
00101
00102     mtk::Tools::BeginUnitTestNo(3);
00103

```

```

00104     mtk::Real aa = 0.0;
00105     mtk::Real bb = 1.0;
00106     mtk::Real cc = 0.0;
00107     mtk::Real dd = 1.0;
00108     mtk::Real ee = 0.0;
00109     mtk::Real ff = 1.0;
00110
00111     int nn = 5;
00112     int mm = 7;
00113     int oo = 6;
00114
00115     mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00116
00117     bool assertion{true};
00118
00119     assertion = assertion && (gg.west_bndy() == aa);
00120     assertion = assertion && (gg.east_bndy() == bb);
00121     assertion = assertion && (gg.num_cells_x() == nn);
00122     assertion = assertion && (gg.south_bndy() == cc);
00123     assertion = assertion && (gg.north_bndy() == dd);
00124     assertion = assertion && (gg.num_cells_y() == mm);
00125     assertion = assertion && (gg.bottom_bndy() == ee);
00126     assertion = assertion && (gg.top_bndy() == ff);
00127     assertion = assertion && (gg.num_cells_z() == oo);
00128
00129     mtk::Tools::EndUnitTestNo(3);
00130     mtk::Tools::Assert(assertion);
00131 }
00132
00133 mtk::Real ScalarField(const mtk::Real &xx,
00134                      const mtk::Real &yy,
00135                      const mtk::Real &zz) {
00136
00137     return xx + yy + zz;
00138 }
00139
00140 void TestBindScalarFieldWriteToFile() {
00141
00142     mtk::Tools::BeginUnitTestNo(4);
00143
00144     mtk::Real aa = 0.0;
00145     mtk::Real bb = 1.0;
00146     mtk::Real cc = 0.0;
00147     mtk::Real dd = 1.0;
00148     mtk::Real ee = 0.0;
00149     mtk::Real ff = 1.0;
00150
00151     int nn = 50;
00152     int mm = 50;
00153     int oo = 50;
00154
00155     mtk::UniStgGrid3D gg(aa, bb, nn, cc, dd, mm, ee, ff, oo);
00156
00157     gg.BindScalarField(ScalarField);
00158
00159     if(!gg.WriteToFile("mtk_uni_stg_grid_3d_test_04.dat",
00160                       "x",
00161                       "y",
00162                       "z",
00163                       "u(x,y,z)")) {
00164         std::cerr << "Error writing to file." << std::endl;
00165     }
00166
00167     mtk::Tools::EndUnitTestNo(4);
00168 }
00169
00170 int main () {
00171
00172     std::cout << "Testing mtk::UniStgGrid3D class." << std::endl;
00173
00174     TestDefaultConstructor();
00175     TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOSTreamOperator();
00176     TestGetters();
00177     TestBindScalarFieldWriteToFile();
00178 }
00179
00180 #else
00181 #include <iostream>
00182 using std::cout;
00183 using std::endl;
00184 int main () {

```

```
00185     cout << "This code HAS to be compiled with support for C++11." << endl;
00186     cout << "Exiting..." << endl;
00187 }
00188 #endif
```

Index

BANDED
Enumerations., [37](#)

COL_MAJOR
Enumerations., [37](#)

CRS
Enumerations., [37](#)

DENSE
Enumerations., [37](#)

Data structures., [39](#)

Enumerations., [36](#)
BANDED, [37](#)
COL_MAJOR, [37](#)
CRS, [37](#)
DENSE, [37](#)
ROW_MAJOR, [37](#)
SCALAR, [36](#)
SCALAR_TO_VECTOR, [36](#)
VECTOR, [36](#)
VECTOR_TO_SCALAR, [36](#)

Execution tools., [38](#)

Grids., [41](#)

Mimetic operators., [42](#)

mtk, [45](#)
operator<<, [48](#), [49](#)

Numerical methods., [40](#)

operator<<
mtk, [48](#), [49](#)

ROW_MAJOR
Enumerations., [37](#)

Real
Roots., [34](#)

Roots., [33](#)
Real, [34](#)

SCALAR
Enumerations., [36](#)

SCALAR_TO_VECTOR
Enumerations., [36](#)

VECTOR