MTK: Mimetic Methods Toolkit

Generated by Doxygen 1.8.8

Contents

Chapter 1

Introduction

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

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

1.1 MTK Concerns

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

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

In order of dependence these are:

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

1.2 MTK Flavors

The MTK collection of wrappers is:

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

Others are being strongly considered.

1.3 Contact, Support and Credits

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

Developers are members of:

1. Mimetic Numerical Methods Research and Development Group.

2. Computational Geoscience Research and Development Group.

3. Ocean Modeling Research and Development Group.

Currently the developers are:

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

2. Jose E. Castillo, Ph.D. - jcastillo at mail dot sdsu dot edu

3. Guillermo F. Miranda, Ph.D. - unigrav at hotmail dot com

4. Christopher P. Paolini, Ph.D. - paolini at engineering dot sdsu dot edu

5. Angel Boada.

6. Johnny Corbino.

7. Raul Vargas–Navarro.

1.4 Acknowledgements and Contributions

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

1. Mohammad Abouali, Ph.D.

2. Dany De Cecchis, Ph.D.

3. Otilio Rojas, Ph.D.

4. Julia Rossi.

Chapter 2

Programming Tools

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

1. Editor: Kate - KDE Advanced Text Editor. Version 3.13.3. Using KDE Development Platform 4.13.3 (C) 2000-2005 The Kate Authors.

2. 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 3

Licensing and Modifications

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

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

1. Modifications to source code should be reported to: `esanchez@mail.sdsu.edu` and a copy of the modified files should be reported once modifications are completed, unless these modifications are made through the project's GitHub page: `http://www.csrc.sdsu.edu/mtk`. Documentation related to said modifications should be developed and included in any deliverable.

2. Redistributions of source code must be done through direct downloads from the project's GitHub page: `http://www.csrc.sdsu.edu/mtk`

3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

4. Usage of the binary form on proprietary applications shall require explicit prior written permission from the the copyright holders, and due credit should be given to the copyright holders.

5. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

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

Chapter 4

Read Me File and Installation Instructions

README File for the Mimetic Methods Toolkit (MTK)

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

1. Description

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

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

2. Dependencies

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

$(HOME)/Libraries/

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

For Linux:

```
1. LAPACK - Available from:  http://www.netlib.org/lapack/
   1. BLAS - Available from:  http://www.netlib.org/blas/

2. GLPK - Available from:  https://www.gnu.org/software/glpk/

3. (Optional) ATLAS - Available from:  http://math-atlas.sourceforge.net/
   1. LAPACK - Available from:  http://www.netlib.org/lapack/
     1. BLAS - Available from:  http://www.netlib.org/blas

4. (Optional) Valgrind - Available from:  http://valgrind.org/

5. (Optional) Doxygen - Available from  http://www.stack.nl/~dimitri/doxygen/
```

For OS X:

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

3. Installation

PART 1. CONFIGURATION OF THE MAKEFILE.

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

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

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

- testall: runs all the tests.

- gendoc: generates the documentation for the library.

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

PART 2. BUILD THE LIBRARY.

```
$ make
```

If successful you'll read (before building the examples):

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

4. Contact, Support, and Credits

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

Developers are members of:

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

Currently the developers are:

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

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

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 "
}
```

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

Thanks and happy coding!

Chapter 5

Tests and Test Architectures

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

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

```
1. 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

Examples

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

Chapter 7

User Manual, References and Theory

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

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

However, a .PDF copy of this manual can be found here.

Chapter 8

Todo List

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

Implement Kronecker product using the BLAS.

Member mtk::DenseMatrix::OrderColMajor ()

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

Member mtk::DenseMatrix::OrderRowMajor ()

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

Member mtk::DenseMatrix::Transpose ()

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

Class mtk::GLPKAdapter

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

Member mtk::Matrix::IncreaseNumNull () 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 UniStg↩ Grid2D &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 UniStg↩ Grid2D &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::Tools::test_number_

Check usage of static methods and private members.

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_div_1d.cc

Overload ostream operator as in mtk::Lap1D.

Implement creation of ■ w. mtk::BLASAdapter.

File mtk_glpk_adapter_test.cc

Test the mtk::GLPKAdapter class.

File mtk_grad_1d.cc

Overload ostream operator as in mtk::Lap1D.

Implement creation of ■ w. mtk::BLASAdapter.

File mtk_lapack_adapter.cc

Write documentation using LaTeX.

File mtk_lapack_adapter_test.cc

Test the mtk::LAPACKAdapter class.

File mtk_quad_1d.h

Implement this class.

File mtk_roots.h

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

Test selective precision mechanisms.

File mtk_uni_stg_grid_1d.h

Create overloaded binding routines that read data from files.

File mtk_uni_stg_grid_2d.h

Create overloaded binding routines that read data from files.

File mtk_uni_stg_grid_3d.h

Create overloaded binding routines that read data from files.

Chapter 9

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 10

Module Index

10.1   Modules

Here is a list of all modules:

Chapter 11

Namespace Index

11.1   Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 12

Class Index

12.1   Class List

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

Chapter 13

File Index

13.1 File List

Here is a list of all files with brief descriptions:

Chapter 14

Module Documentation

14.1 Roots.

Fundamental execution parameters and defined types.

Typedefs

- typedef float mtk::Real

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

Variables

- const float mtk::kZero {0.0f}

    MTK's zero defined according to selective compilation.
- const float mtk::kOne {1.0f}

    MTK's one defined according to selective compilation.
- const float mtk::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 int mtk::kDefaultOrderAccuracy {2}

    Default order of accuracy for mimetic operators.
- const float mtk::kDefaultMimeticThreshold {1e-6f}

    Default tolerance for higher-order mimetic operators.
- const int mtk::kCriticalOrderAccuracyDiv {8}

    At this order (and higher) we must use the CBSA to construct.
- const int mtk::kCriticalOrderAccuracyGrad {10}

    At this order (and higher) we must use the CBSA to construct.

14.1.1 Detailed Description

Fundamental execution parameters and defined types.

14.1.2 Typedef Documentation

14.1.2.1 mtk::Real

Definition at line 83 of file mtk_roots.h.

14.1.3 Variable Documentation

14.1.3.1 mtk::kCriticalOrderAccuracyDiv {8}

Definition at line 167 of file mtk_roots.h.

14.1.3.2 mtk::kCriticalOrderAccuracyGrad {10}

Definition at line 176 of file mtk_roots.h.

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

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 157 of file mtk_roots.h.

14.1.3.4 mtk::kDefaultOrderAccuracy {2}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 143 of file mtk_roots.h.

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

Definition at line 131 of file mtk_roots.h.

14.1.3.6 mtk::kOne {1.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 117 of file mtk_roots.h.

14.1.3.7 mtk::kTwo {2.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 118 of file mtk_roots.h.

14.1.3.8   mtk::kZero {0.0f}

Warning

Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 116 of file mtk_roots.h.

14.2 Enumerations.

Enumerations.

Enumerations

- enum mtk::MatrixStorage { mtk::DENSE, mtk::BANDED, mtk::CRS }

    Considered matrix storage schemes to implement sparse matrices.

- enum mtk::MatrixOrdering { mtk::ROW_MAJOR, mtk::COL_MAJOR }

    Considered matrix ordering (for Fortran purposes).

- enum mtk::FieldNature { mtk::SCALAR, mtk::VECTOR }

    Nature of the field discretized in a given grid.

- enum mtk::DirInterp { mtk::SCALAR_TO_VECTOR, mtk::VECTOR_TO_SCALAR }

    Interpolation operator.

14.2.1 Detailed Description

Enumerations.

14.2.2 Enumeration Type Documentation

14.2.2.1 enum mtk::DirInterp

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.

14.2.2.2 enum mtk::FieldNature

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

See also

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

Enumerator

SCALAR	Scalar-valued field.
VECTOR	Vector-valued field.

Definition at line 113 of file mtk_enums.h.

14.2.2.3 enum **mtk::MatrixOrdering**

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

See also

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

Enumerator

> ***ROW_MAJOR*** Row-major ordering (C/C++).
>
> ***COL_MAJOR*** Column-major ordering (Fortran).

Definition at line 95 of file mtk_enums.h.

14.2.2.4 enum **mtk::MatrixStorage**

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

Enumerator

> ***DENSE*** Dense matrices, implemented as a 1D array: DenseMatrix.
>
> ***BANDED*** Banded matrices ala LAPACK and ScaLAPACK: Must be implemented.
>
> ***CRS*** Compressed-Rows Storage: Must be implemented.

Definition at line 77 of file mtk_enums.h.

14.3 Execution tools.

Tools to ensure execution correctness.

Classes

- class mtk::Tools

  Tool manager class.

14.3.1 Detailed Description

Tools to ensure execution correctness.

14.4 Data structures.

Fundamental data structures.

Classes

- class mtk::DenseMatrix

  Defines a common dense matrix, using a 1D array.
- class mtk::Matrix

  Definition of the representation of a matrix in the MTK.

14.4.1 Detailed Description

Fundamental data structures.

14.5 Numerical methods.

Adapter classes and auxiliary numerical methods.

Classes

- class mtk::BLASAdapter

    Adapter class for the BLAS API.
- class mtk::GLPKAdapter

    Adapter class for the GLPK API.
- class mtk::LAPACKAdapter

    Adapter class for the LAPACK API.

14.5.1 Detailed Description

Adapter classes and auxiliary numerical methods.

14.6 Grids.

Uniform rectangular staggered grids.

Classes

- class mtk::UniStgGrid1D

    Uniform 1D Staggered Grid.
- class mtk::UniStgGrid2D

    Uniform 2D Staggered Grid.
- class mtk::UniStgGrid3D

    Uniform 3D Staggered Grid.

14.6.1 Detailed Description

Uniform rectangular staggered grids.

14.7 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($\ast$ mtk::CoefficientFunction0D)(const Real &tt)

    A function of a BC coefficient evaluated on a 0D domain and time.
- typedef Real($\ast$ mtk::CoefficientFunction1D)(const Real &xx, const Real &tt)

    A function of a BC coefficient evaluated on a 1D domain and time.
- typedef Real($\ast$ mtk::CoefficientFunction2D)(const Real &xx, const Real &yy, const Real &tt)

    A function of a BC coefficient evaluated on a 2D domain and time.

14.7.1 Detailed Description

Mimetic operators.

14.7.2 Typedef Documentation

14.7.2.1 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.

14.7.2.2 mtk::CoefficientFunction1D

Definition at line 97 of file mtk_robin_bc_descriptor_2d.h.

14.7.2.3 mtk::CoefficientFunction2D

Definition at line 97 of file mtk_robin_bc_descriptor_3d.h.

Chapter 15

Namespace Documentation

15.1 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 { DENSE, BANDED, CRS }

    Considered matrix storage schemes to implement sparse matrices.

- enum MatrixOrdering { ROW_MAJOR, COL_MAJOR }

    Considered matrix ordering (for Fortran purposes).

- enum FieldNature { SCALAR, VECTOR }

    Nature of the field discretized in a given grid.

- enum DirInterp { SCALAR_TO_VECTOR, VECTOR_TO_SCALAR }

    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 int kDefaultOrderAccuracy {2}

    Default order of accuracy for mimetic operators.
- const float kDefaultMimeticThreshold {1e-6f}

    Default tolerance for higher-order mimetic operators.
- const int kCriticalOrderAccuracyDiv {8}

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

    At this order (and higher) we must use the CBSA to construct.

15.1.1 Function Documentation

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

1. Print approximating coefficients for the interior.

Definition at line 66 of file mtk_interp_1d.cc.

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

1. Print spatial coordinates.

2. Print scalar field.

Definition at line 67 of file mtk_uni_stg_grid_3d.cc.

15.1.1.3 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.

15.1.1.4 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.

15.1.1.5 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.

15.1.1.6 std::ostream& mtk::operator<< (std::ostream & *stream,* mtk::DenseMatrix & *in*)

Definition at line 77 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
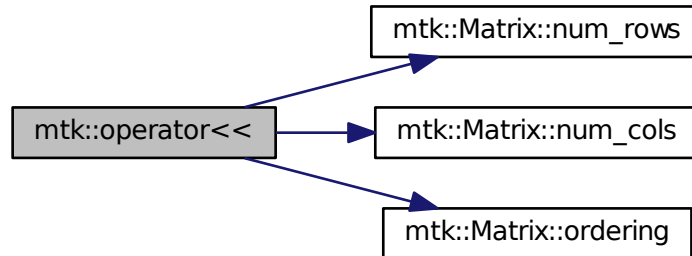
15.1.1.7 std::ostream& mtk::operator $<<$ **(std::ostream &** *stream,* **mtk::Grad1D &** *in* **)**

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 79 of file mtk_grad_1d.cc.

15.1.1.8 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 79 of file mtk_div_1d.cc.

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

Here is the caller graph for this function:
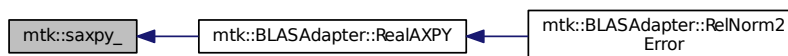
15.1.1.10   void mtk::sgels_ (char ∗ *trans,* int ∗ *m,* int ∗ *n,* int ∗ *nrhs,* Real ∗ *a,* int ∗ *lda,* Real ∗ *b,* int ∗ *ldb,* Real ∗ *work,* int ∗ *lwork,* int ∗ *info*)

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

The following options are provided:

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

   ```
   minimize || B - A*X ||.
   ```

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

3. If TRANS = 'T' and m $>=$ n: find the minimum norm solution of an undetermined system A∗∗T ∗ X = B.

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

   ```
   minimize || B - A**T * X ||.
   ```

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

See also

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

Parameters

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

Here is the caller graph for this function:
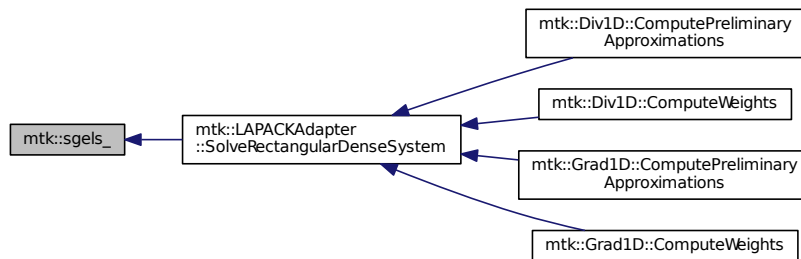
15.1.1.11 void mtk::sgemm_ (char ∗ *transa,* char ∗ *transb,* int ∗ *m,* int ∗ *n,* int ∗ *k,* double ∗ *alpha,* double ∗ *a,* int ∗ *lda,* double ∗ *b,* aamm int ∗ *ldb,* double ∗ *beta,* double ∗ *c,* int ∗ *ldc*)

Here is the caller graph for this function:
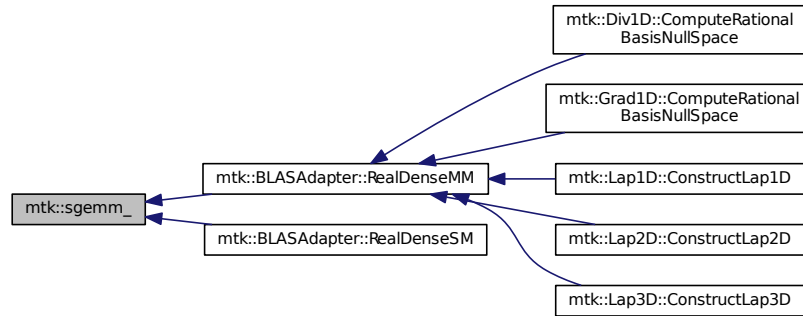
15.1.1.12 void mtk::sgemv_ (char ∗ *trans,* int ∗ *m,* int ∗ *n,* float ∗ *alpha,* float ∗ *a,* int ∗ *lda,* float ∗ *x,* int ∗ *incx,* float ∗ *beta,* float ∗ *y,* int ∗ *incy*)

Here is the caller graph for this function:

15.1.1.13 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 C ∗ Q TRANS = 'T': Q∗∗T ∗ C C ∗ Q∗∗T

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

```
  Q = H(1) H(2) . . . 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	m	The number of columns of the matrix a. n $>=$ 0.
in	n	The number of columns of the matrix a. n $>=$ 0.
in,out	a	On entry, the n-by-n matrix a.
in	lda	Leading dimension matrix. LDA $>=$ max(1,M).
in,out	tau	Scalars from elementary reflectors. min(M,N).
in,out	work	Workspace. info = 0, work(1) is optimal lwork.
in	lwork	The dimension of work. lwork $>=$ max(1,n).
in	info	info = 0: successful exit.

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

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

Here is the caller graph for this function:
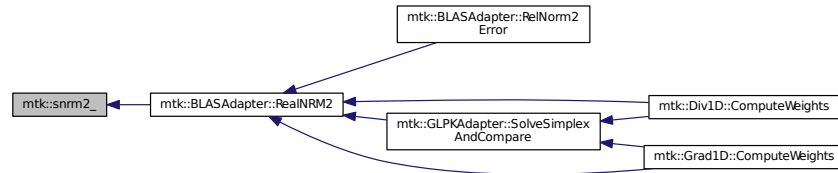
15.1.1.16   void mtk::sormqr_ (char $*$ *side*, char $*$ *trans*, int $*$ *m*, int $*$ *n*, int $*$ *k*, Real $*$ *a*, int $*$ *lda*, Real $*$ *tau*, Real $*$ *c*, int $*$ *ldc*, Real $*$ *work*, int $*$ *lwork*, int $*$ *info*)

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

```
        SIDE = 'L'      SIDE = 'R'
```

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

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

```
  Q = H(1) H(2) . . . 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	side	See Table 1 above.
in	trans	See Table 1 above.
in	m	Number of rows of the C matrix.
in	n	Number of columns of the C matrix.
in	k	Number of reflectors.
in,out	a	The matrix containing the reflectors.
in	lda	The dimension of work. lwork $>=$ max(1,n).
in	tau	Scalar factors of the elementary reflectors.
in	c	Output matrix.
in	ldc	Leading dimension of the output matrix.
in,out	work	Workspace. info = 0, work(1) optimal lwork.
in	lwork	The dimension of work.
in,out	info	info = 0: successful exit.

Chapter 16

Class Documentation

16.1   mtk::BLASAdapter Class Reference

Adapter class for the BLAS API.

`#include <mtk_blas_adapter.h>`

Collaboration diagram for mtk::BLASAdapter:

```
┌─────────────────────────┐
│   mtk::BLASAdapter       │
├─────────────────────────┤
│                         │
├─────────────────────────┤
│ + RealNRM2()            │
│ + RealAXPY()            │
│ + RelNorm2Error()       │
│ + RealDenseMV()         │
│ + RealDenseMM()         │
│ + RealDenseSM()         │
└─────────────────────────┘
```
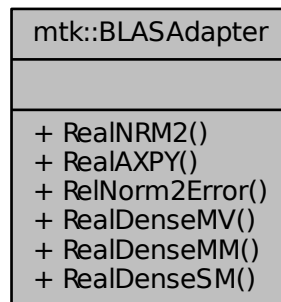
Static Public Member Functions

- static Real RealNRM2 (Real ∗in, int &in_length)

    Compute the $||\mathbf{x}||_2$ of given array $\mathbf{x}$.
- 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.

16.1.1 Detailed Description

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

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

See also

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

Definition at line 96 of file mtk_blas_adapter.h.

16.1.2 Member Function Documentation

16.1.2.1   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

`in`	*alpha*	Scalar of the first array.
`in`	*xx*	First array.
`in`	*yy*	Second array.
`in`	*in_length*	Lengths of the given arrays.

Returns

> Norm-2 of the given array.

Definition at line 339 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
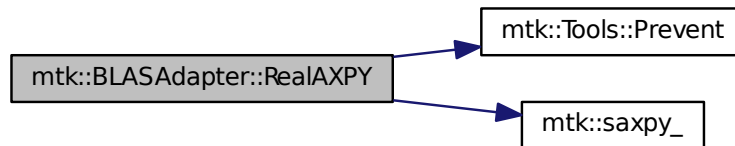
Here is the caller graph for this function:
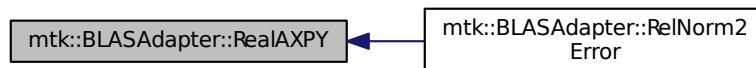
16.1.2.2 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM (mtk::DenseMatrix & *aa,* mtk::DenseMatrix & *bb*) [static]

Performs:

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

Parameters

in	*aa*	First matrix.
in	*bb*	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 409 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
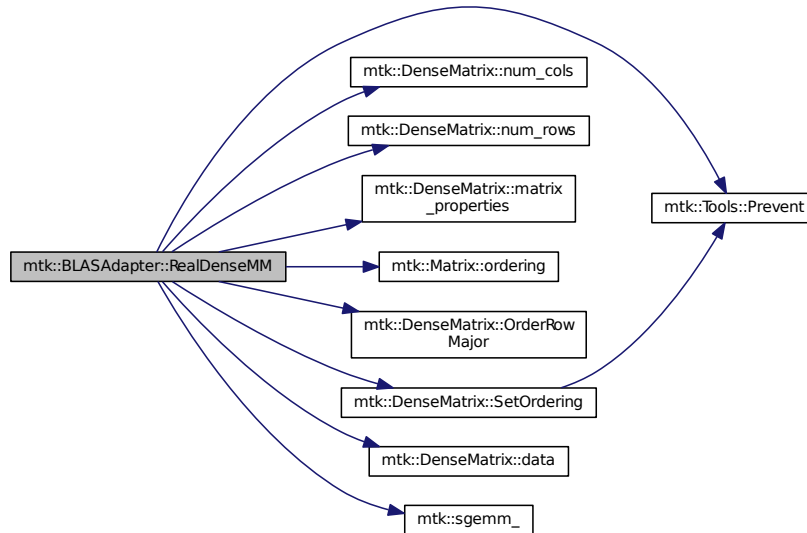
Here is the caller graph for this function:
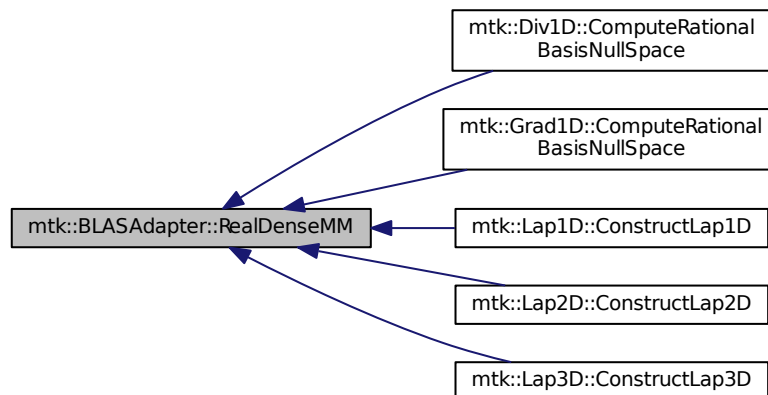
16.1.2.3 void mtk::BLASAdapter::RealDenseMV (mtk::Real & *alpha,* **mtk::DenseMatrix &** *aa,* **mtk::Real** ∗ *xx,* **mtk::Real &** *beta,* **mtk::Real** ∗ *yy* **)** `[static]`

Performs

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

Parameters

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

See also

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

Definition at line 378 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
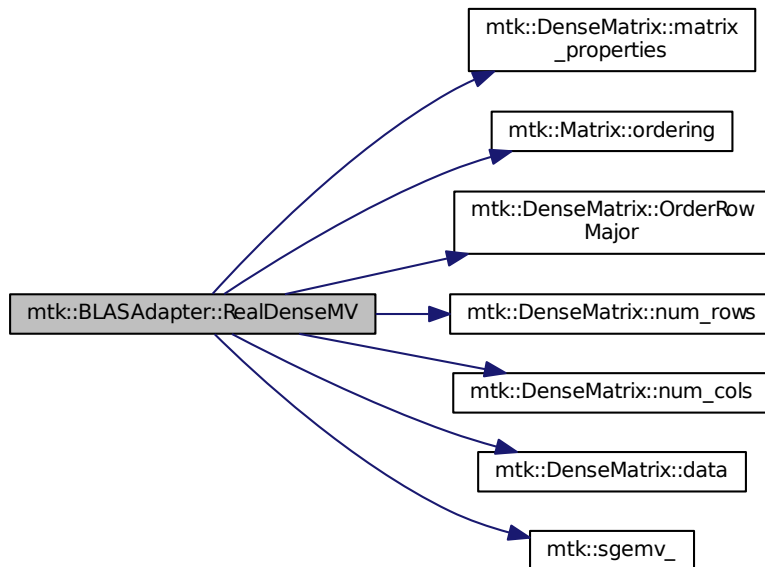
Here is the caller graph for this function:
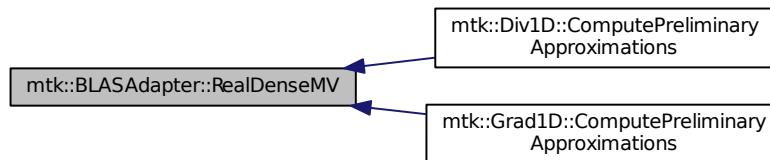
16.1.2.4 mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM (mtk::Real *alpha,* **mtk::DenseMatrix &** *aa* **)** `[static]`

Performs:

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

Parameters

in	*alpha*	Input scalar.
in	*aa*	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 466 of file mtk_blas_adapter.cc.

Here is the call graph for this function:

```
mtk::BLASAdapter::RealDenseSM ──→ mtk::Tools::Prevent
                              ──→ mtk::DenseMatrix::num_rows
                              ──→ mtk::DenseMatrix::num_cols
                              ──→ mtk::DenseMatrix::matrix_properties
                              ──→ mtk::Matrix::ordering
                              ──→ mtk::DenseMatrix::OrderRowMajor
                              ──→ mtk::DenseMatrix::data
                              ──→ mtk::sgemm_
```
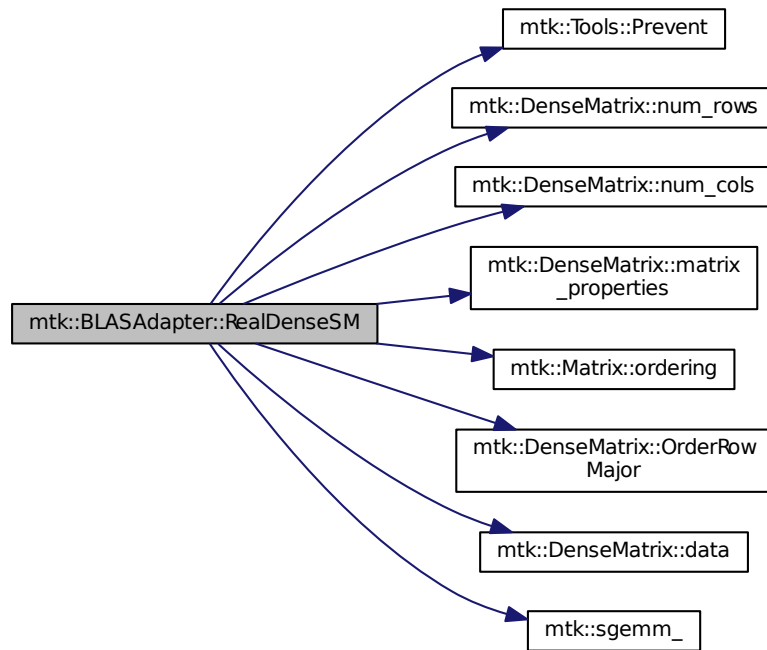
16.1.2.5 mtk::Real mtk::BLASAdapter::RealNRM2 (Real ∗ *in,* int & *in_length*) `[static]`

Parameters

in	*in*	Input array.
in	*in_length*	Length of the array.

Returns

>   Norm-2 of the given array.

Definition at line 324 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
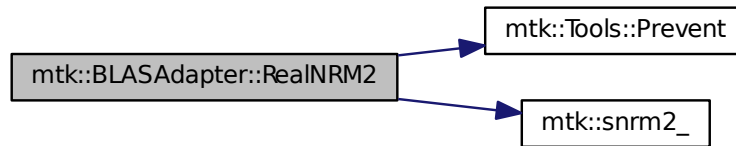
Here is the caller graph for this function:
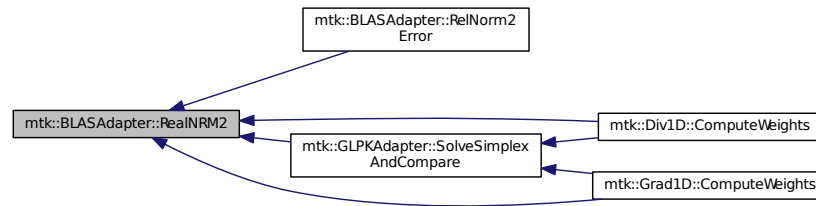
16.1.2.6 mtk::Real mtk::BLASAdapter::RelNorm2Error (mtk::Real ∗ *computed,* mtk::Real ∗ *known,* int *length*) `[static]`

We compute

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

Parameters

in	*known*	Array containing the computed solution.
in	*computed*	Array containing the known solution (ref. solution).

Returns

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

Definition at line 358 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
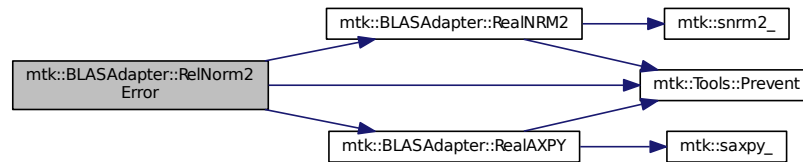
The documentation for this class was generated from the following files:

- include/mtk_blas_adapter.h

- src/mtk_blas_adapter.cc

16.2 mtk::Curl2D Class Reference

Implements a 2D mimetic curl operator.

`#include <mtk_curl_2d.h>`

Collaboration diagram for mtk::Curl2D:
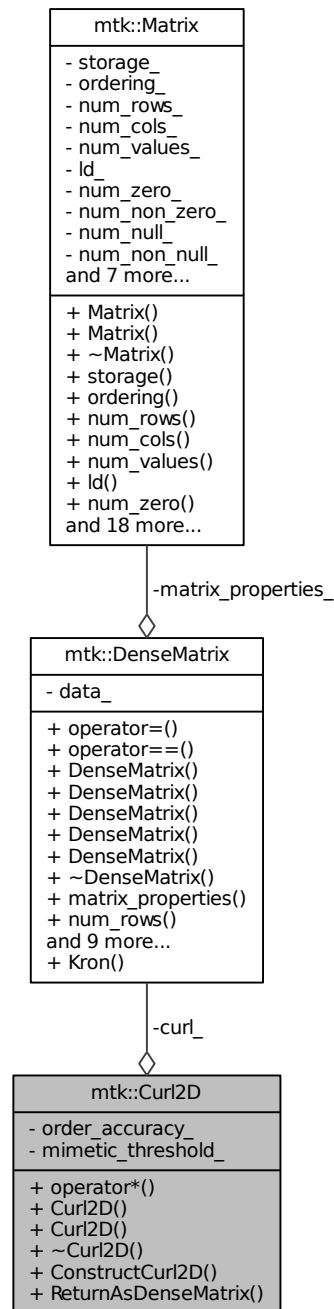
Public Member Functions

- 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.

16.2.1 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.

16.2.2 Constructor & Destructor Documentation

16.2.2.1 mtk::Curl2D::Curl2D ( )

Definition at line 79 of file mtk_curl_2d.cc.

16.2.2.2 mtk::Curl2D::Curl2D (const Curl2D & *curl*)

Parameters

in	*curl*	Given curl.

Definition at line 83 of file mtk_curl_2d.cc.

16.2.2.3 mtk::Curl2D::∼Curl2D ( )

Definition at line 87 of file mtk_curl_2d.cc.

16.2.3   Member Function Documentation

16.2.3.1   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:
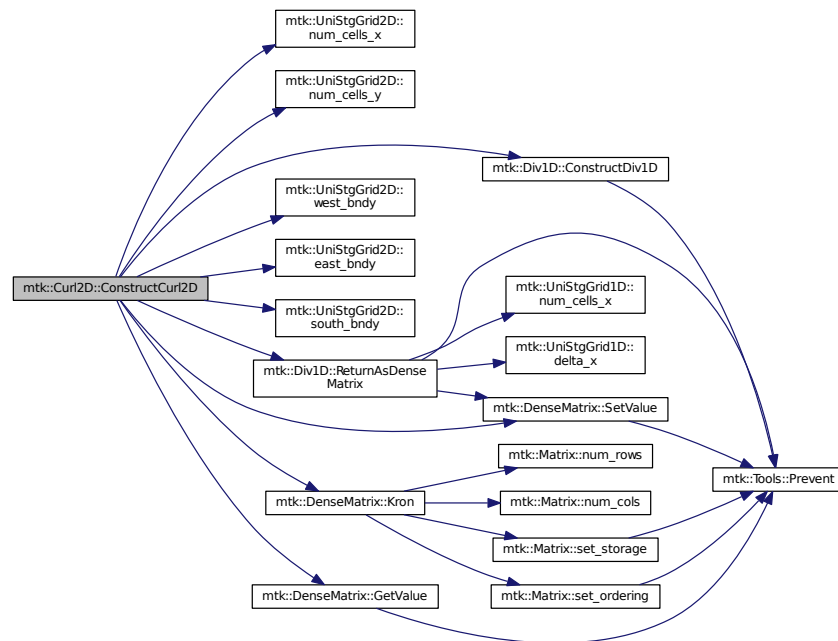
16.2.3.2   mtk::UniStgGrid3D mtk::Curl2D::operator∗ (const UniStgGrid2D & *grid* **) const**

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

Definition at line 70 of file mtk_curl_2d.cc.

16.2.3.3   mtk::DenseMatrix mtk::Curl2D::ReturnAsDenseMatrix (   ) const

Returns

> The operator as a dense matrix.

Definition at line 157 of file mtk_curl_2d.cc.

16.2.4   Member Data Documentation

16.2.4.1   **DenseMatrix mtk::Curl2D::curl_** `[private]`

Definition at line 112 of file mtk_curl_2d.h.

16.2.4.2   **Real mtk::Curl2D::mimetic_threshold_** `[private]`

Definition at line 116 of file mtk_curl_2d.h.

16.2.4.3   **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

16.3   **mtk::DenseMatrix Class Reference**

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

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

Collaboration diagram for mtk::DenseMatrix:

```
┌─────────────────────────┐
│      mtk::Matrix        │
├─────────────────────────┤
│ - storage_              │
│ - ordering_             │
│ - num_rows_             │
│ - num_cols_             │
│ - num_values_           │
│ - ld_                   │
│ - num_zero_             │
│ - num_non_zero_         │
│ - num_null_             │
│ - num_non_null_         │
│ and 7 more...           │
├─────────────────────────┤
│ + Matrix()              │
│ + Matrix()              │
│ + ~Matrix()             │
│ + storage()             │
│ + ordering()            │
│ + num_rows()            │
│ + num_cols()            │
│ + num_values()          │
│ + ld()                  │
│ + num_zero()            │
│ and 18 more...          │
└─────────────────────────┘
            │
            │ -matrix_properties_
            ◇
┌─────────────────────────┐
│    mtk::DenseMatrix     │
├─────────────────────────┤
│ - data_                 │
├─────────────────────────┤
│ + operator=()           │
│ + operator==()          │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ and 9 more...           │
│ + Kron()                │
└─────────────────────────┘
```

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).

16.3.1   Detailed Description

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

Definition at line 92 of file mtk_dense_matrix.h.

16.3.2   Constructor & Destructor Documentation

16.3.2.1   mtk::DenseMatrix::DenseMatrix (   )

Definition at line 162 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
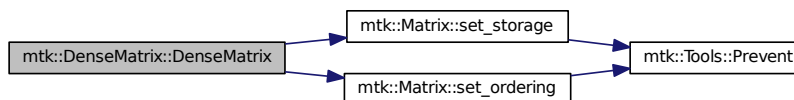
16.3.2.2   mtk::DenseMatrix::DenseMatrix (const **DenseMatrix** & *in*)

Parameters

`in`	*in*	Given matrix.

Definition at line 168 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
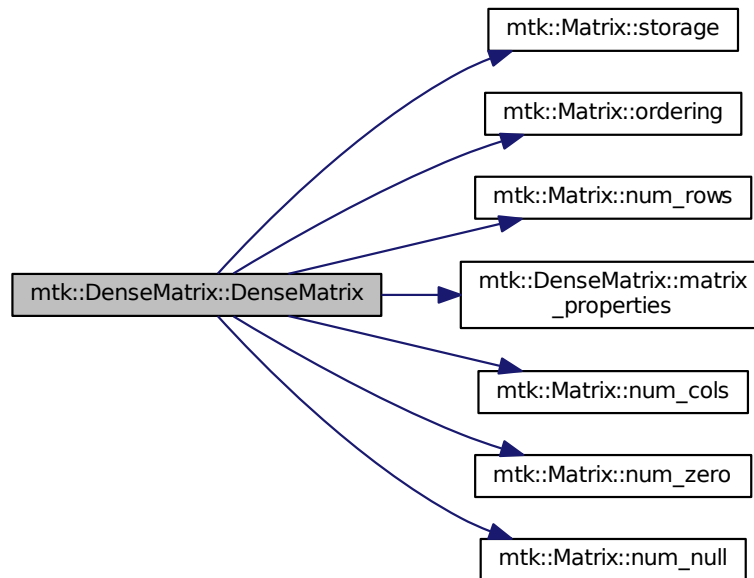
16.3.2.3   mtk::DenseMatrix::DenseMatrix (const int & *num_rows,* const int & *num_cols*)

Parameters

in	*num_rows*	Number of rows of the required matrix.
in	*num_cols*	Number of rows of the required matrix.

Exceptions

std::bad_alloc	

Definition at line 201 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
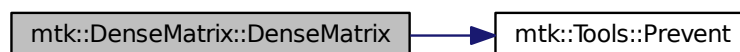
16.3.2.4   mtk::DenseMatrix::DenseMatrix (const int & *rank,* **const bool &** *padded,* **const bool &** *transpose* **)**

Used in the construction of the mimetic operators.

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

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

Parameters

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

Exceptions

std::bad_alloc	

Definition at line 223 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

16.3.2.5   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 in 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 nor the CRS nor the CBS algorithms may work for irregular geometries, such as curvilinear grids.

Parameters

in	*gen*	Given generator vector.
in	*gen_length*	Length generator vector.
in	*pro_length*	Length the progression.
in	*transpose*	Should the transpose be created instead?

Exceptions

std::bad_alloc	

Definition at line 264 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

16.3.2.6 mtk::DenseMatrix::∼DenseMatrix ( )

Definition at line 312 of file mtk_dense_matrix.cc.

16.3.3 Member Function Documentation

16.3.3.1 mtk::Real ∗ mtk::DenseMatrix::data ( ) const `[noexcept]`

Returns

Pointer to an array of mtk::Real.

Definition at line 343 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
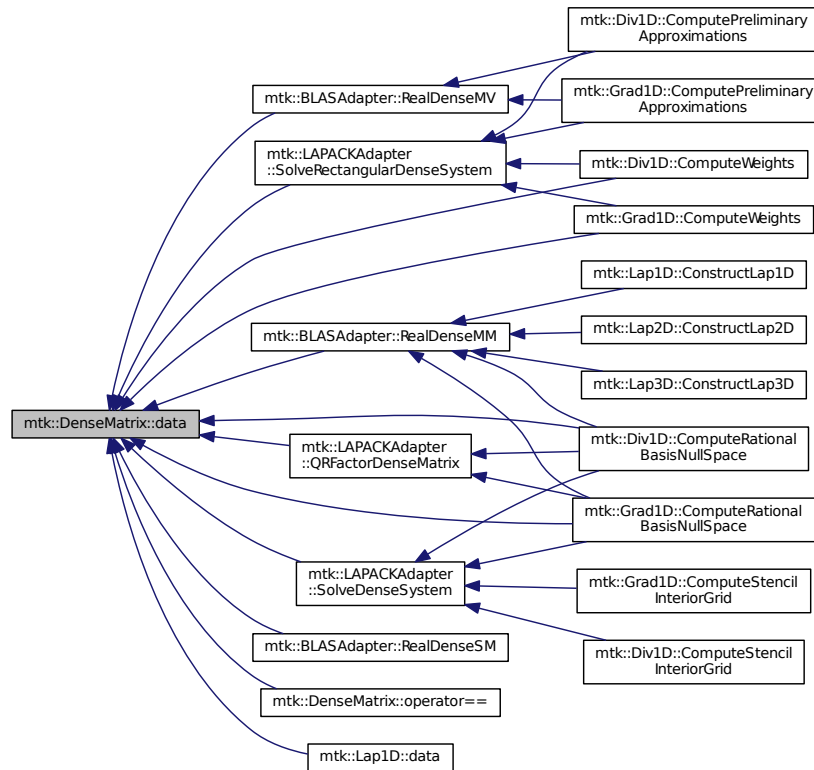
16.3.3.2 mtk::Real mtk::DenseMatrix::GetValue (const int & *row_coord,* const int & *col_coord*) const `[noexcept]`

Parameters

in	*row_coord*	Row coordinate.
in	*col_coord*	Column coordinate.

Returns

> The required value at the specified coordinates.

Definition at line 348 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

| mtk::DenseMatrix::GetValue | → | mtk::Tools::Prevent |

Here is the caller graph for this function:

```
                              mtk::Curl2D::ConstructCurl2D

                              mtk::Div2D::ConstructDiv2D
                                                            mtk::Lap2D::ConstructLap2D
                              mtk::Grad2D::ConstructGrad2D

                              mtk::Div3D::ConstructDiv3D
                                                            mtk::Lap3D::ConstructLap3D
                              mtk::Grad3D::ConstructGrad3D

 mtk::DenseMatrix::GetValue   mtk::Lap1D::ConstructLap1D

                              mtk::Lap1D::ReturnAsDense
                              Matrix

                              mtk::RobinBCDescriptor1D
                              ::ImposeOnLaplacianMatrix

                              mtk::RobinBCDescriptor2D
                              ::ImposeOnWestBoundaryNoSpace

                              mtk::RobinBCDescriptor2D
                              ::ImposeOnEastBoundaryNoSpace
```

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

Parameters

in	*aa*	First matrix.

in	*bb*	Second matrix.

Exceptions

std::bad_alloc	

Todo Implement Kronecker product using the BLAS.

Definition at line 490 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
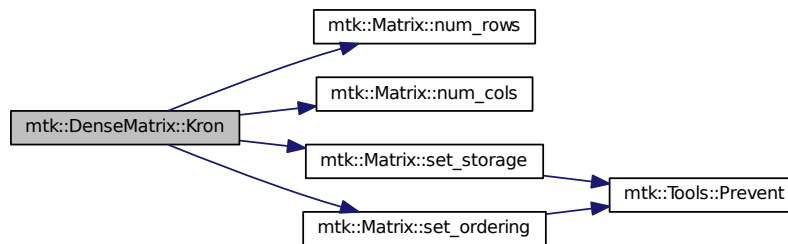
Here is the caller graph for this function:
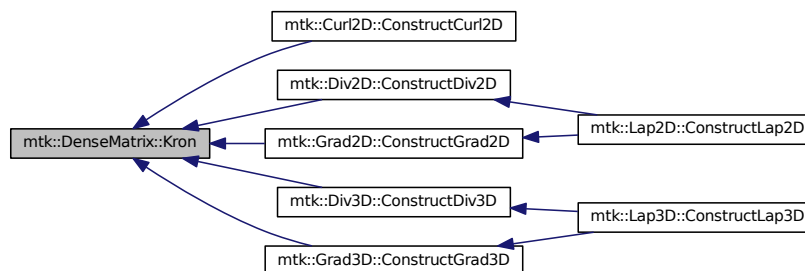
16.3.3.4 mtk::Matrix mtk::DenseMatrix::matrix_properties ( ) const `[noexcept]`

Returns

Pointer to a [Matrix](#).

Definition at line 318 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
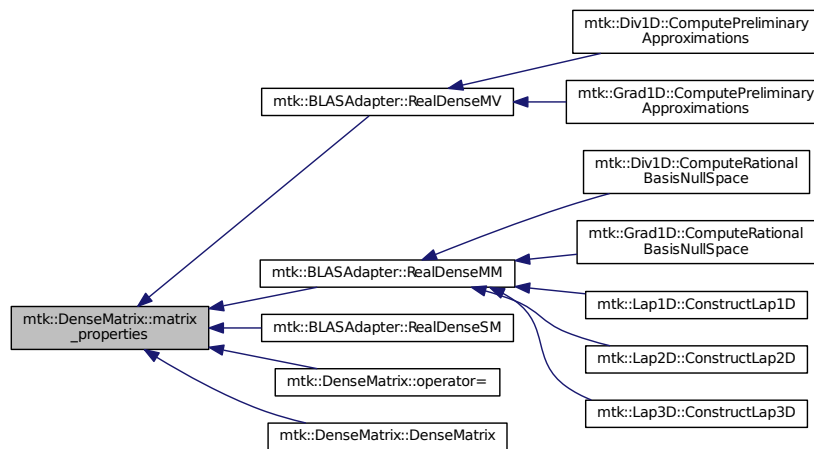
16.3.3.5 int mtk::DenseMatrix::num_cols ( ) const `[noexcept]`

Returns

Number of columns of the matrix.

Definition at line 338 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
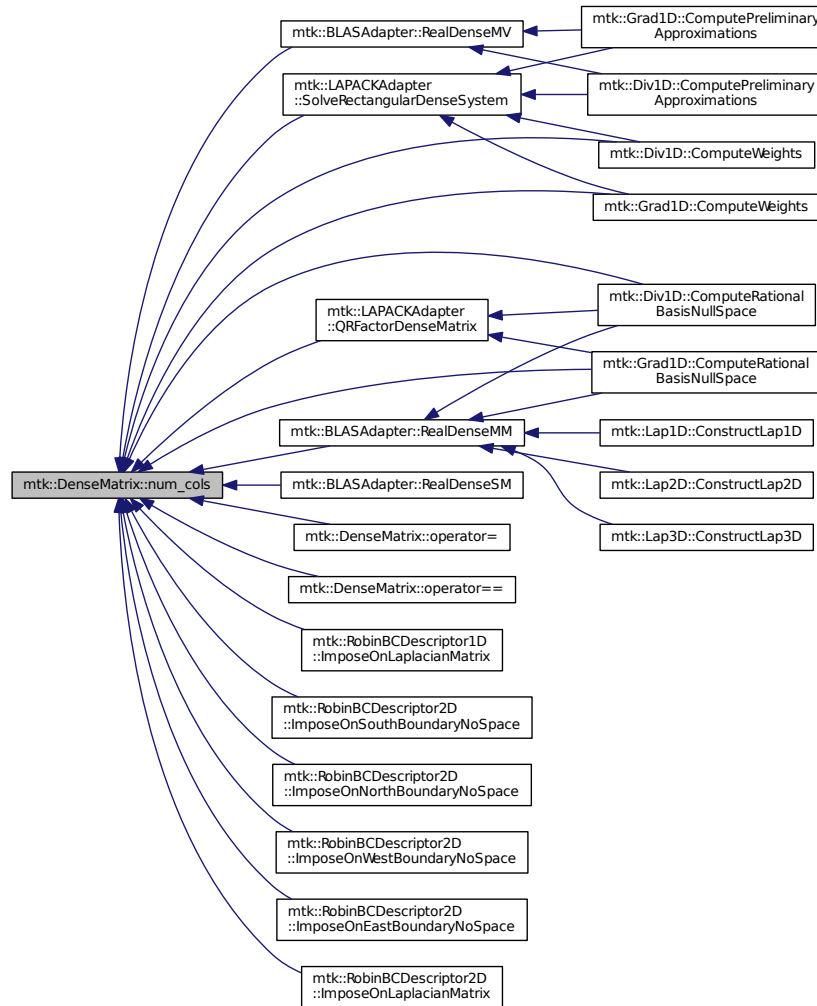
16.3.3.6 int mtk::DenseMatrix::num_rows ( ) const `[noexcept]`

Returns

Number of rows of the matrix.

Definition at line 333 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
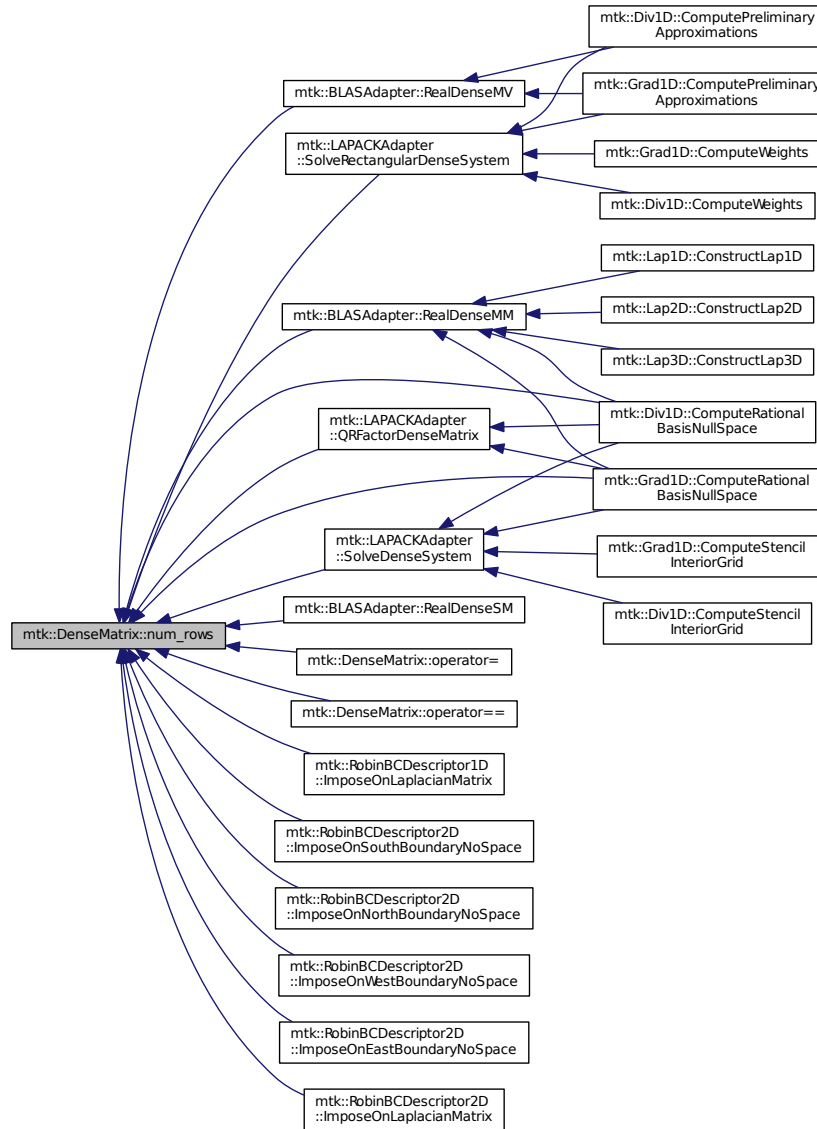
16.3.3.7   mtk::DenseMatrix & mtk::DenseMatrix::operator= (const DenseMatrix & *in*)

Parameters

in		*in*	Given matrix.

Returns

Copy of the given matrix.

Definition at line 100 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
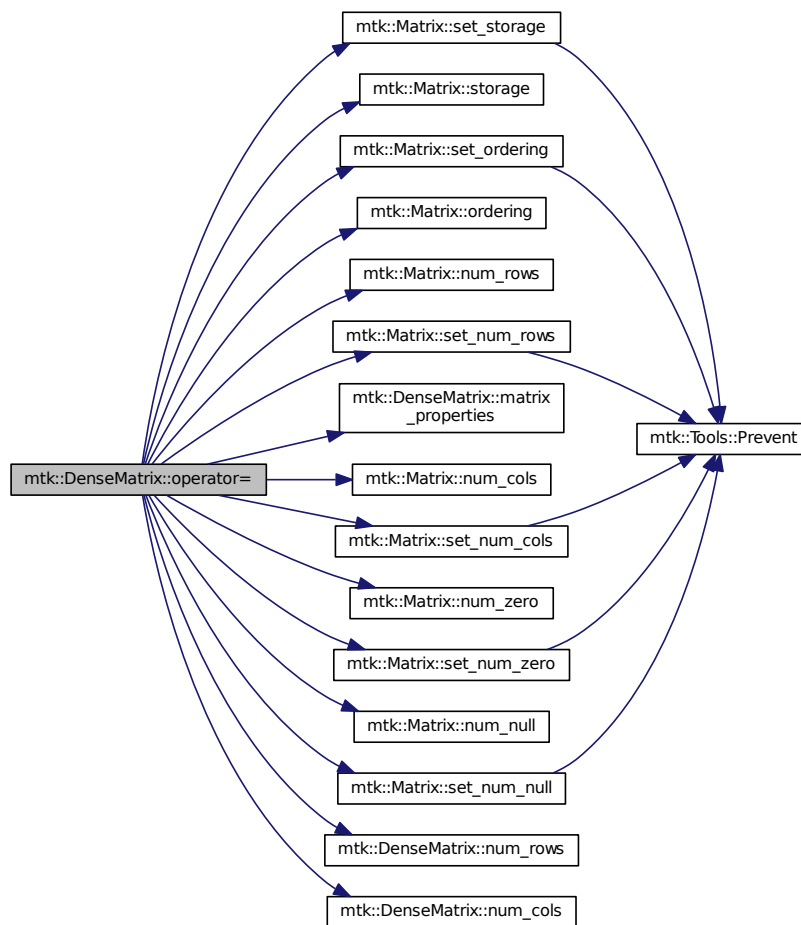
16.3.3.8   bool mtk::DenseMatrix::operator== (const DenseMatrix & *in*)

Definition at line 141 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
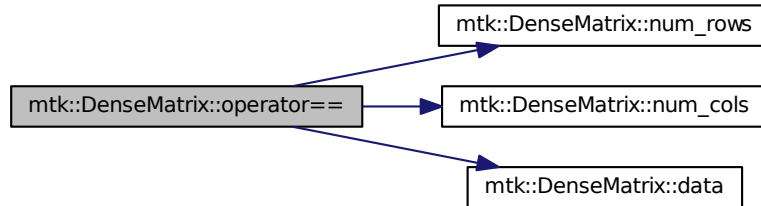
16.3.3.9 void mtk::DenseMatrix::OrderColMajor ( )

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

Definition at line 451 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
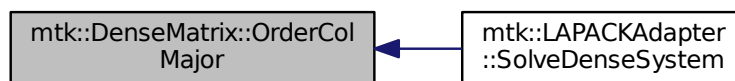
16.3.3.10 void mtk::DenseMatrix::OrderRowMajor ( )

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

Definition at line 410 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
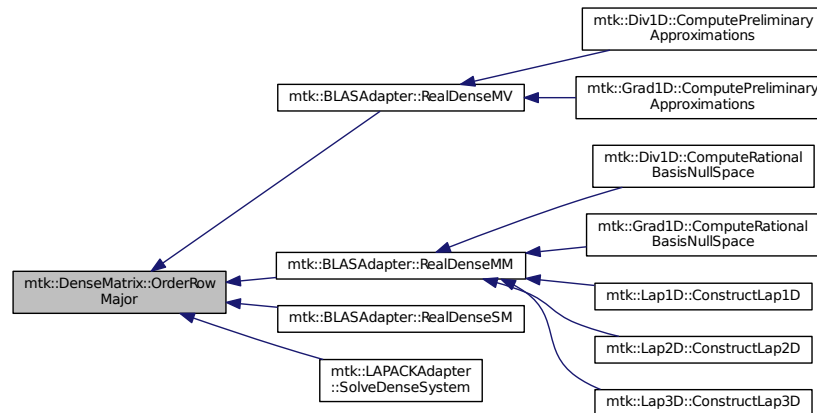
16.3.3.11 void mtk::DenseMatrix::SetOrdering (mtk::MatrixOrdering *oo*) `[noexcept]`

Parameters

`in`	*oo*	Ordering.

Returns

The required value at the specified coordinates.

Definition at line 323 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
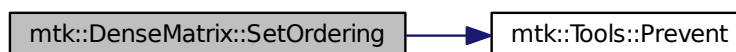
Here is the caller graph for this function:
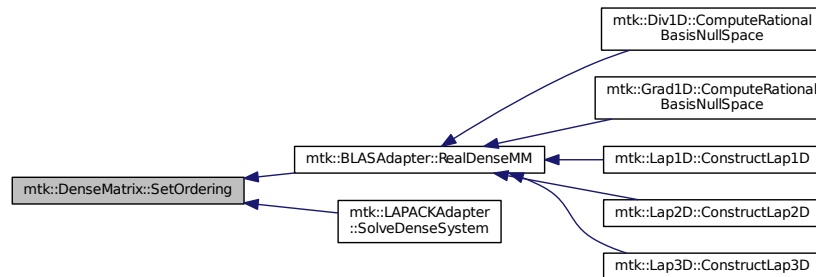
16.3.3.12 void mtk::DenseMatrix::SetValue (const int & *row_coord,* const int & *col_coord,* const **Real & *val*)** `[noexcept]`

Parameters

in	*row_coord*	Row coordinate.
in	*col_coord*	Column coordinate.
in	*val*	Row Actual value to be inserted.

Definition at line 360 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

mtk::Curl2D::ConstructCurl2D

mtk::Div1D::ReturnAsDense
Matrix

mtk::Div2D::ConstructDiv2D

mtk::Div3D::ConstructDiv3D

mtk::Lap2D::ConstructLap2D

mtk::Lap1D::ConstructLap1D

mtk::Grad1D::ReturnAsDense
Matrix

mtk::Grad2D::ConstructGrad2D

mtk::Lap3D::ConstructLap3D

mtk::Grad3D::ConstructGrad3D

mtk::Div1D::mim_bndy

mtk::Grad1D::mim_bndy

mtk::RobinBCDescriptor1D
::ImposeOnLaplacianMatrix

mtk::DenseMatrix::SetValue

mtk::Grad1D::ReturnAsDimensionless
DenseMatrix

mtk::Interp1D::ReturnAsDense
Matrix

mtk::Lap1D::ReturnAsDense
Matrix

mtk::RobinBCDescriptor2D
::ImposeOnSouthBoundaryNoSpace

mtk::RobinBCDescriptor2D
::ImposeOnSouthBoundaryWithSpace

mtk::RobinBCDescriptor2D
::ImposeOnNorthBoundaryNoSpace

mtk::RobinBCDescriptor2D
::ImposeOnNorthBoundaryWithSpace

mtk::RobinBCDescriptor2D
::ImposeOnWestBoundaryNoSpace

mtk::RobinBCDescriptor2D
::ImposeOnWestBoundaryWithSpace

mtk::RobinBCDescriptor2D
::ImposeOnEastBoundaryNoSpace

mtk::RobinBCDescriptor2D
::ImposeOnEastBoundaryWithSpace

16.3.3.13   void mtk::DenseMatrix::Transpose (   )

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

Definition at line 373 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:

```
                                    ┌──────────────────────────┐
                                    │ mtk::Div1D::ComputeRational│
                                    │      BasisNullSpace       │
                                    └──────────────────────────┘
                                    ┌──────────────────────────┐
                                    │ mtk::Div1D::ComputeWeights │
                                    └──────────────────────────┘
    ┌──────────────────────────┐
    │ mtk::DenseMatrix::Transpose│
    └──────────────────────────┘
                                    ┌──────────────────────────┐
                                    │ mtk::Grad1D::ComputeRational│
                                    │      BasisNullSpace       │
                                    └──────────────────────────┘
                                    ┌──────────────────────────┐
                                    │ mtk::Grad1D::ComputeWeights │
                                    └──────────────────────────┘
```

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

Parameters

in	*filename*	Name of the output file.

Returns

> Success of the file writing process.

See also

> http://www.gnuplot.info/

Definition at line 531 of file mtk_dense_matrix.cc.

16.3.4   Friends And Related Function Documentation

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

Definition at line 77 of file mtk_dense_matrix.cc.

16.3.5   Member Data Documentation

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

Definition at line 291 of file mtk_dense_matrix.h.

16.3.5.2 **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

16.4 mtk::Div1D Class Reference

Implements a 1D mimetic divergence operator.

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

Collaboration diagram for mtk::Div1D:

```
                    ┌─────────────────────┐
                    │     mtk::Matrix     │
                    ├─────────────────────┤
                    │ - storage_          │
                    │ - ordering_         │
                    │ - num_rows_         │
                    │ - num_cols_         │
                    │ - num_values_       │
                    │ - ld_               │
                    │ - num_zero_         │
                    │ - num_non_zero_     │
                    │ - num_null_         │
                    │ - num_non_null_     │
                    │ and 7 more...       │
                    ├─────────────────────┤
                    │ + Matrix()          │
                    │ + Matrix()          │
                    │ + ~Matrix()         │
                    │ + storage()         │
                    │ + ordering()        │
                    │ + num_rows()        │
                    │ + num_cols()        │
                    │ + num_values()      │
                    │ + ld()              │
                    │ + num_zero()        │
                    │ and 18 more...      │
                    └─────────────────────┘
                              ◇ -matrix_properties_
                    ┌─────────────────────┐
                    │  mtk::DenseMatrix   │
                    ├─────────────────────┤
                    │ - data_             │
                    ├─────────────────────┤
                    │ + operator=()       │
                    │ + operator==()      │
                    │ + DenseMatrix()     │
                    │ + DenseMatrix()     │
                    │ + DenseMatrix()     │
                    │ + DenseMatrix()     │
                    │ + DenseMatrix()     │
                    │ + ~DenseMatrix()    │
                    │ + matrix_properties()│
                    │ + num_rows()        │
                    │ and 9 more...       │
                    │ + Kron()            │
                    └─────────────────────┘
                              ◇ -rat_basis_null_space_
                    ┌─────────────────────────────────┐
                    │           mtk::Div1D            │
                    ├─────────────────────────────────┤
                    │ - order_accuracy_               │
                    │ - dim_null_                     │
                    │ - num_bndy_coeffs_              │
                    │ - divergence_length_            │
                    │ - minrow_                       │
                    │ - row_                          │
                    │ - coeffs_interior_              │
                    │ - prem_apps_                    │
                    │ - weights_crs_                  │
                    │ - weights_cbs_                  │
                    │ - mim_bndy_                     │
                    │ - divergence_                   │
                    │ - mimetic_threshold_            │
                    ├─────────────────────────────────┤
                    │ + Div1D()                       │
                    │ + Div1D()                       │
                    │ + ~Div1D()                      │
                    │ + ConstructDiv1D()              │
                    │ + num_bndy_coeffs()             │
                    │ + coeffs_interior()             │
                    │ + weights_crs()                 │
                    │ + weights_cbs()                 │
                    │ + mim_bndy()                    │
                    │ + ReturnAsDenseMatrix()         │
                    │ - ComputeStencilInteriorGrid()  │
                    │ - ComputeRationalBasisNull      │
                    │ Space()                         │
                    │ - ComputePreliminaryApproximations()│
                    │ - ComputeWeights()              │
                    │ - ComputeStencilBoundaryGrid()  │
                    │ - AssembleOperator()            │
                    └─────────────────────────────────┘
```

Public Member Functions

- Div1D ()

Default constructor.

- Div1D (const Div1D &div)

    Copy constructor.

- ∼Div1D ()

    Destructor.

- bool ConstructDiv1D (int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimetic↩
  Threshold)

    Factory method implementing the CBS Algorithm to build operator.

- int num_bndy_coeffs () const

    Returns how many coefficients are approximating at the boundary.

- Real ∗ coeffs_interior () const

    Returns coefficients for the interior of the grid.

- Real ∗ weights_crs (void) const

    Return collection of weights as computed by the CRSA.

- Real ∗ weights_cbs (void) const

    Return collection of weights as computed by the CBSA.

- DenseMatrix mim_bndy () const

    Return collection of mimetic approximations at the boundary.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid) const

    Return the operator as a dense matrix.

Private Member Functions

- bool ComputeStencilInteriorGrid (void)

    Stage 1 of the CBS Algorithm.

- bool ComputeRationalBasisNullSpace (void)

    Stage 2.1 of the CBS Algorithm.

- bool ComputePreliminaryApproximations (void)

    Stage 2.2 of the CBS Algorithm.

- bool ComputeWeights (void)

    Stage 2.3 of the CBS Algorithm.

- bool ComputeStencilBoundaryGrid (void)

    Stage 2.4 of the CBS Algorithm.

- bool AssembleOperator (void)

    Stage 3 of the CBS Algorithm.

Private Attributes

- int order_accuracy_

    Order of numerical accuracy of the operator.

- int dim_null_

    Dim. null-space for boundary approximations.

- int num_bndy_coeffs_

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

- int divergence_length_

    Length of the output array.

- int minrow_

> *Row from the optimizer with the minimum rel. nor.*

- int row_

    Row currently processed by the optimizer.

- DenseMatrix rat_basis_null_space_

    Rational b. null-space w. bndy.

- Real * coeffs_interior_

    Interior stencil.

- Real * prem_apps_

    2D array of boundary preliminary approximations.

- Real * weights_crs_

    Array containing weights from CRSA.

- Real * weights_cbs_

    Array containing weights from CBSA.

- Real * mim_bndy_

    Array containing mimetic boundary approximations.

- Real * divergence_

    Output array containing the operator and weights.

- Real mimetic_threshold_

    $<$ *Mimetic threshold.*

Friends

- std::ostream & operator$<<$ (std::ostream &stream, Div1D &in)

    Output stream operator for printing.

16.4.1   Detailed Description

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

Definition at line 81 of file mtk_div_1d.h.

16.4.2   Constructor & Destructor Documentation

16.4.2.1   mtk::Div1D::Div1D (   )

Definition at line 125 of file mtk_div_1d.cc.

16.4.2.2   mtk::Div1D::Div1D (const Div1D & *div*)

Parameters

in	*div*	Given divergence.

Definition at line 140 of file mtk_div_1d.cc.

16.4.2.3   mtk::Div1D::∼Div1D (   )

Definition at line 155 of file mtk_div_1d.cc.

16.4.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

1. The first entry of the array will contain the order of accuracy.

2. The second entry the collection of coefficients for interior of grid.

3. If order_accuracy_ > 2, then third entry is the collection of weights.

4. If order_accuracy_ > 2, next dim_null_ entries is approximating coefficients for the west boundary of the grid.

Definition at line 1342 of file mtk_div_1d.cc.

16.4.3.2 mtk::Real ∗ mtk::Div1D::coeffs_interior ( ) const

Returns

Coefficients for the interior of the grid.

Definition at line 320 of file mtk_div_1d.cc.

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

Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.

2. Compute the dim_null near-the-boundary columns of the pi matrix.

3. Create the Vandermonde matrix for this iteration.

4. New order-selector vector (gets re-written with LAPACK solutions).

5. Solving TT∗rr = ob yields the columns rr of the KK matrix.

6. Scale the KK matrix to make it a rational basis for null-space.

7. Extract the last dim_null values of the pre-scaled ob.

8. Once we posses the bottom elements, we proceed with the scaling.

Definition at line 691 of file mtk_div_1d.cc.

Here is the call graph for this function:
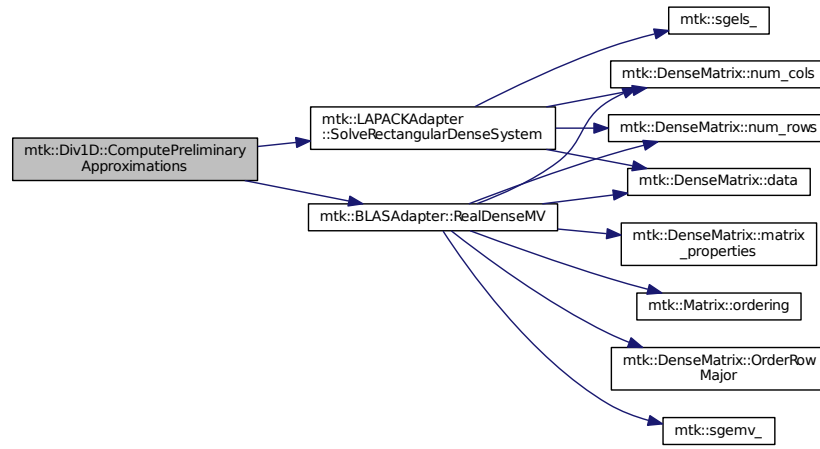
16.4.3.4   bool mtk::Div1D::ComputeRationalBasisNullSpace (void) `[private]`

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

1. Create generator vector for the first approximation.

2. Create Vandermonde matrix.

3. QR-factorize the Vandermonde matrix.

4. Extract the basis for the null-space from Q matrix.

5. Scale null-space to make it rational.

Definition at line 515 of file mtk_div_1d.cc.

Here is the call graph for this function:
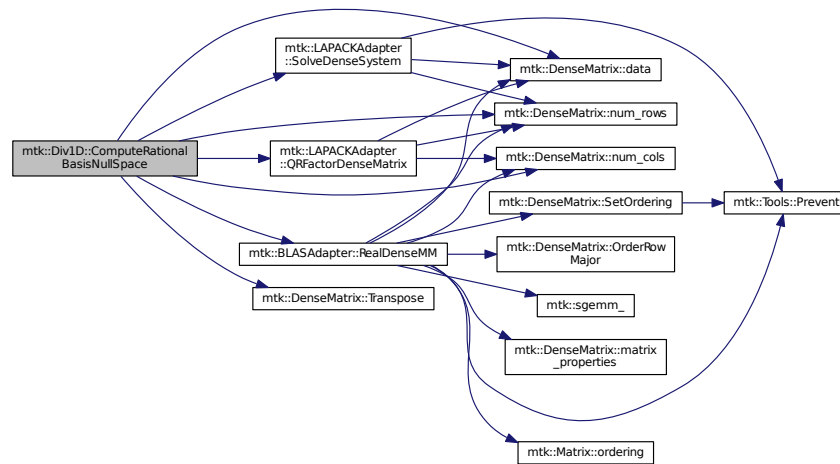
16.4.3.5   bool mtk::Div1D::ComputeStencilBoundaryGrid (void) `[private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.

2. Compute alpha values.

3. Compute the mimetic boundary approximations.

Definition at line 1241 of file mtk_div_1d.cc.

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

Compute the stencil approximating the interior of the staggered grid.

1. Create vector for interior spatial coordinates.

2. Create Vandermonde matrix (using interior coordinates as generator).

3. Create order-selector vector.

4. Solve dense Vandermonde system to attain the interior coefficients.

Definition at line 414 of file mtk_div_1d.cc.

Here is the call graph for this function:
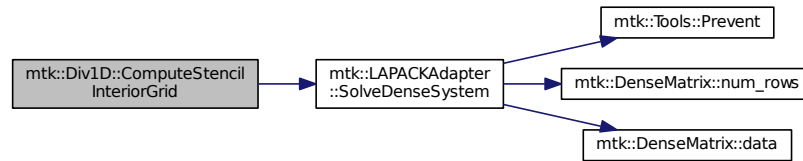
16.4.3.7   bool mtk::Div1D::ComputeWeights (void) `[private]`

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

1. Construct the ■ matrix.

2. Use interior stencil to build proper RHS vector **h**.

3. Get weights (as **CRSA**): ■**q** = **h**.

4. If required order is greater than critical order, start the **CBSA**.

5. Create ■ matrix from ■.

6. Prepare constraint vector as in the CBSA: ■.

7. Brute force search through all the rows of the Φ matrix.

8. Apply solution found from brute force search.

Definition at line 911 of file mtk_div_1d.cc.

Here is the call graph for this function:
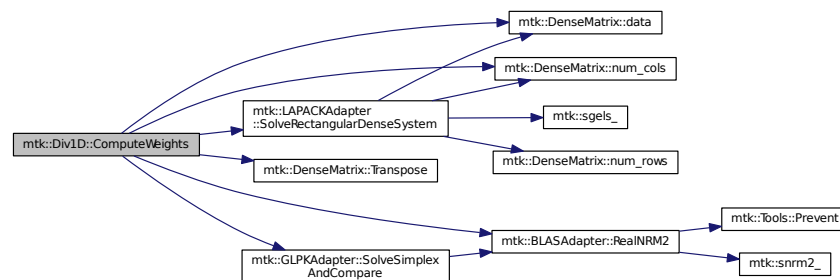
16.4.3.8 bool mtk::Div1D::ConstructDiv1D (int *order_accuracy* = kDefaultOrderAccuracy, mtk::Real *mimetic_threshold* = kDefaultMimeticThreshold)

Returns

Success of the construction.

1. Compute stencil for the interior cells.

2. Compute a rational basis for the null-space for the first matrix.

3. Compute preliminary approximation (non-mimetic) on the boundaries.

4. Compute quadrature weights to impose the mimetic conditions.

5. Compute real approximation (mimetic) on the boundaries.

6. Assemble operator.

Definition at line 176 of file mtk_div_1d.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

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

Returns

>   Collection of mimetic approximations at the boundary.

Definition at line 335 of file mtk_div_1d.cc.

Here is the call graph for this function:

```
┌─────────────────────┐      ┌───────────────────────────┐      ┌──────────────────────┐
│ mtk::Div1D::mim_bndy│─────▶│ mtk::DenseMatrix::SetValue │─────▶│ mtk::Tools::Prevent  │
└─────────────────────┘      └───────────────────────────┘      └──────────────────────┘
```

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

Returns

>   How many coefficients are approximating at the boundary.

Definition at line 315 of file mtk_div_1d.cc.

16.4.3.11   mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix (const UniStgGrid1D & *grid*) 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 350 of file mtk_div_1d.cc.

Here is the call graph for this function:

```
                                      ┌───────────────────────┐
                                      │ mtk::UniStgGrid1D::    │
                                  ┌──▶│    num_cells_x         │
                                  │   └───────────────────────┘
┌─────────────────────┐          │                              ┌──────────────────────┐
│ mtk::Div1D::ReturnAsDense│──────┤                         ┌───▶│ mtk::Tools::Prevent  │
│         Matrix       │──────┐   │   ┌───────────────────┐ │    └──────────────────────┘
└─────────────────────┘      │   └──▶│ mtk::UniStgGrid1D::│ │
                             │       │    delta_x          │ │
                             │       └───────────────────┘ │
                             │   ┌───────────────────────────┐
                             └──▶│ mtk::DenseMatrix::SetValue │
                                 └───────────────────────────┘
```

Here is the caller graph for this function:

mtk::Div1D::ReturnAsDense Matrix	→	mtk::Curl2D::ConstructCurl2D
		mtk::Div2D::ConstructDiv2D	mtk::Lap2D::ConstructLap2D
		mtk::Div3D::ConstructDiv3D	mtk::Lap3D::ConstructLap3D
		mtk::Lap1D::ConstructLap1D

16.4.3.12   mtk::Real ∗ mtk::Div1D::weights_cbs (void) const

Returns

Collection of weights as computed by the CBSA.

Definition at line 330 of file mtk_div_1d.cc.

16.4.3.13   mtk::Real ∗ mtk::Div1D::weights_crs (void) const

Returns

Collection of weights as computed by the CRSA.

Definition at line 325 of file mtk_div_1d.cc.

16.4.4   Friends And Related Function Documentation

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

1. Print order of accuracy.

2. Print approximating coefficients for the interior.

3. Print mimetic weights.

4. Print mimetic approximations at the boundary.

Definition at line 79 of file mtk_div_1d.cc.

16.4.5   Member Data Documentation

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

Definition at line 202 of file mtk_div_1d.h.

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

Definition at line 194 of file mtk_div_1d.h.

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

Definition at line 207 of file mtk_div_1d.h.

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

Definition at line 196 of file mtk_div_1d.h.

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

Definition at line 206 of file mtk_div_1d.h.

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

Definition at line 209 of file mtk_div_1d.h.

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

Definition at line 197 of file mtk_div_1d.h.

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

Definition at line 195 of file mtk_div_1d.h.

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

Definition at line 193 of file mtk_div_1d.h.

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

Definition at line 203 of file mtk_div_1d.h.

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

Definition at line 200 of file mtk_div_1d.h.

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

Definition at line 198 of file mtk_div_1d.h.

16.4.5.13 Real∗ mtk::Div1D::weights_cbs_ `[private]`

Definition at line 205 of file mtk_div_1d.h.

16.4.5.14 Real∗ mtk::Div1D::weights_crs_ `[private]`

Definition at line 204 of file mtk_div_1d.h.

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

- include/mtk_div_1d.h

- src/mtk_div_1d.cc

16.5 mtk::Div2D Class Reference

Implements a 2D mimetic divergence operator.

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

Collaboration diagram for mtk::Div2D:

```
┌─────────────────────────┐
│      mtk::Matrix        │
├─────────────────────────┤
│ - storage_              │
│ - ordering_             │
│ - num_rows_             │
│ - num_cols_             │
│ - num_values_           │
│ - ld_                   │
│ - num_zero_             │
│ - num_non_zero_         │
│ - num_null_             │
│ - num_non_null_         │
│ and 7 more...           │
├─────────────────────────┤
│ + Matrix()              │
│ + Matrix()              │
│ + ~Matrix()             │
│ + storage()             │
│ + ordering()            │
│ + num_rows()            │
│ + num_cols()            │
│ + num_values()          │
│ + ld()                  │
│ + num_zero()            │
│ and 18 more...          │
└─────────────────────────┘
            │
            │ -matrix_properties_
            ◇
┌─────────────────────────┐
│    mtk::DenseMatrix     │
├─────────────────────────┤
│ - data_                 │
├─────────────────────────┤
│ + operator=()           │
│ + operator==()          │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ and 9 more...           │
│ + Kron()                │
└─────────────────────────┘
            │
            │ -divergence_
            ◇
┌─────────────────────────┐
│      mtk::Div2D         │
├─────────────────────────┤
│ - order_accuracy_       │
│ - mimetic_threshold_    │
├─────────────────────────┤
│ + Div2D()               │
│ + Div2D()               │
│ + ~Div2D()              │
│ + ConstructDiv2D()      │
│ + ReturnAsDenseMatrix() │
└─────────────────────────┘
```

Public Member Functions

- Div2D ()

Default constructor.

- Div2D (const Div2D &div)

    Copy constructor.

- ∼Div2D ()

    Destructor.

- bool ConstructDiv2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_↩
    threshold=kDefaultMimeticThreshold)

    Factory method implementing the CBS Algorithm to build operator.

- DenseMatrix ReturnAsDenseMatrix () const

    Return the operator as a dense matrix.

Private Attributes

- DenseMatrix divergence_

    Actual operator.

- int order_accuracy_

    Order of accuracy.

- Real mimetic_threshold_

    Mimetic Threshold.

16.5.1 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.

16.5.2 Constructor & Destructor Documentation

16.5.2.1 mtk::Div2D::Div2D ( )

Definition at line 69 of file mtk_div_2d.cc.

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

Parameters

in	*div*	Given divergence.

Definition at line 73 of file mtk_div_2d.cc.

16.5.2.3 mtk::Div2D::∼Div2D ( )

Definition at line 77 of file mtk_div_2d.cc.

16.5.3 Member Function Documentation

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

Returns

Success of the construction.

Definition at line 79 of file mtk_div_2d.cc.

Here is the call graph for this function:
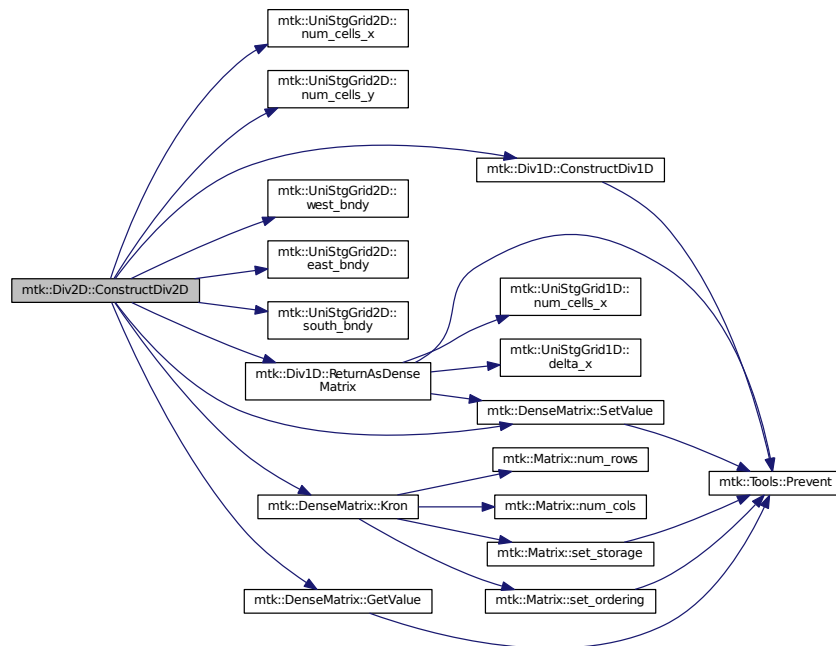
Here is the caller graph for this function:

16.5.3.2 **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:

16.5.4 Member Data Documentation

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

Definition at line 108 of file mtk_div_2d.h.

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

Definition at line 112 of file mtk_div_2d.h.

16.5.4.3 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

16.6 mtk::Div3D Class Reference

Implements a 3D mimetic divergence operator.

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

Collaboration diagram for mtk::Div3D:

```
                          ┌─────────────────────┐
                          │     mtk::Matrix     │
                          ├─────────────────────┤
                          │ - storage_          │
                          │ - ordering_         │
                          │ - num_rows_         │
                          │ - num_cols_         │
                          │ - num_values_       │
                          │ - ld_               │
                          │ - num_zero_         │
                          │ - num_non_zero_     │
                          │ - num_null_         │
                          │ - num_non_null_     │
                          │ and 7 more...       │
                          ├─────────────────────┤
                          │ + Matrix()          │
                          │ + Matrix()          │
                          │ + ~Matrix()         │
                          │ + storage()         │
                          │ + ordering()        │
                          │ + num_rows()        │
                          │ + num_cols()        │
                          │ + num_values()      │
                          │ + ld()              │
                          │ + num_zero()        │
                          │ and 18 more...      │
                          └─────────────────────┘
                                    │ -matrix_properties_
                                    ◇
                          ┌─────────────────────┐
                          │  mtk::DenseMatrix   │
                          ├─────────────────────┤
                          │ - data_             │
                          ├─────────────────────┤
                          │ + operator=()       │
                          │ + operator==()      │
                          │ + DenseMatrix()     │
                          │ + DenseMatrix()     │
                          │ + DenseMatrix()     │
                          │ + DenseMatrix()     │
                          │ + DenseMatrix()     │
                          │ + ~DenseMatrix()    │
                          │ + matrix_properties()│
                          │ + num_rows()        │
                          │ and 9 more...       │
                          │ + Kron()            │
                          └─────────────────────┘
                                    │ -divergence_
                                    ◇
                          ┌─────────────────────┐
                          │     mtk::Div3D      │
                          ├─────────────────────┤
                          │ - order_accuracy_   │
                          │ - mimetic_threshold_│
                          ├─────────────────────┤
                          │ + Div3D()           │
                          │ + Div3D()           │
                          │ + ~Div3D()          │
                          │ + ConstructDiv3D()  │
                          │ + ReturnAsDenseMatrix()│
                          └─────────────────────┘
```

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.

16.6.1 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.

16.6.2 Constructor & Destructor Documentation

16.6.2.1 mtk::Div3D::Div3D ( )

Definition at line 67 of file mtk_div_3d.cc.

16.6.2.2 mtk::Div3D::Div3D (const Div3D & *div*)

Parameters

in	*div*	Given divergence.

Definition at line 71 of file mtk_div_3d.cc.

16.6.2.3 mtk::Div3D::∼Div3D ( )

Definition at line 75 of file mtk_div_3d.cc.

16.6.3 Member Function Documentation

16.6.3.1 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:
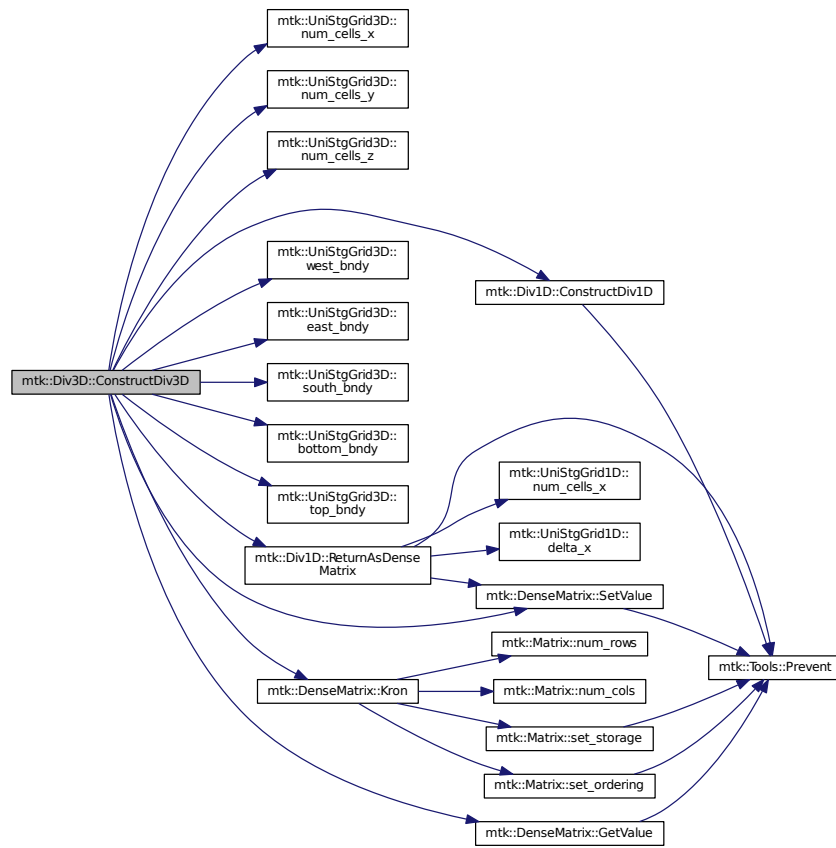
Here is the caller graph for this function:

16.6.3.2 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:

16.6.4 Member Data Documentation

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

Definition at line 108 of file mtk_div_3d.h.

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

Definition at line 112 of file mtk_div_3d.h.

16.6.4.3 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

16.7 mtk::GLPKAdapter Class Reference

Adapter class for the GLPK API.

`#include <mtk_glpk_adapter.h>`

Collaboration diagram for mtk::GLPKAdapter:
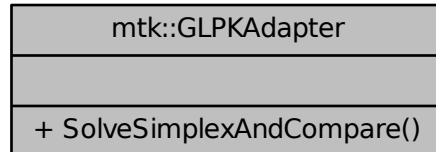
Static Public Member Functions

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

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

16.7.1 Detailed Description

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

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

Warning

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

See also

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

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

Definition at line 101 of file mtk_glpk_adapter.h.

16.7.2 Member Function Documentation

16.7.2.1 **mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare (mtk::Real ∗ *A,* int *nrows,* int *ncols,* int *kk,* mtk::Real ∗** **_hh,_ mtk::Real ∗ *qq,* int *robjective,* mtk::Real *mimetic_tol,* int *copy*)** `[static]`

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

Parameters

in	*alpha*	First scalar.
in	*AA*	Given matrix.
in	*xx*	First vector.
in	*beta*	Second scalar.
in	*beta*	Second scalar.
in,out	*yy*	Second vector (output).
in	*xx*	First vector.
in	*beta*	Second scalar.
in	*beta*	Second scalar.

Returns

Relative error computed between attained solution and provided ref.

Warning

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

1. Memory allocation.

2. Fill the problem.

3. Copy the row to the vector objective.

4. Forming the RHS.

5. Setting up the objective function.

6. Setting up constraints.

7. Copy the matrix minus the row objective to the glpk problem.

8. Solve problem.

Definition at line 76 of file mtk_glpk_adapter.cc.

Here is the call graph for this function:

Here is the caller graph for this function:
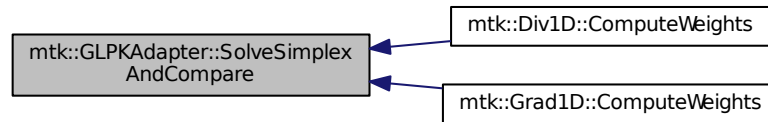
The documentation for this class was generated from the following files:

- include/mtk_glpk_adapter.h

- src/mtk_glpk_adapter.cc

16.8 mtk::Grad1D Class Reference

Implements a 1D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad1D:

```
┌─────────────────────────┐
│      mtk::Matrix        │
├─────────────────────────┤
│ - storage_              │
│ - ordering_             │
│ - num_rows_             │
│ - num_cols_             │
│ - num_values_           │
│ - ld_                   │
│ - num_zero_             │
│ - num_non_zero_         │
│ - num_null_             │
│ - num_non_null_         │
│ and 7 more...           │
├─────────────────────────┤
│ + Matrix()              │
│ + Matrix()              │
│ + ~Matrix()             │
│ + storage()             │
│ + ordering()            │
│ + num_rows()            │
│ + num_cols()            │
│ + num_values()          │
│ + ld()                  │
│ + num_zero()            │
│ and 18 more...          │
└─────────────────────────┘
           │ -matrix_properties_
           ◇
┌─────────────────────────┐
│   mtk::DenseMatrix      │
├─────────────────────────┤
│ - data_                 │
├─────────────────────────┤
│ + operator=()           │
│ + operator==()          │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ and 9 more...           │
│ + Kron()                │
└─────────────────────────┘
           │ -rat_basis_null_space_
           ◇
┌─────────────────────────────────┐
│       mtk::Grad1D               │
├─────────────────────────────────┤
│ - order_accuracy_               │
│ - dim_null                      │
│ - num_bndy_approxs_             │
│ - num_bndy_coeffs_              │
│ - gradient_length_              │
│ - minrow_                       │
│ - row_                          │
│ - coeffs_interior_              │
│ - prem_apps_                    │
│ - weights_crs_                  │
│ - weights_cbs_                  │
│ - mim_bndy_                     │
│ - gradient_                     │
│ - mimetic_threshold_            │
├─────────────────────────────────┤
│ + Grad1D()                      │
│ + Grad1D()                      │
│ + ~Grad1D()                     │
│ + ConstructGrad1D()             │
│ + num_bndy_coeffs()             │
│ + coeffs_interior()             │
│ + weights_crs()                 │
│ + weights_cbs()                 │
│ + mim_bndy()                    │
│ + ReturnAsDenseMatrix()         │
│ + ReturnAsDenseMatrix()         │
│ + ReturnAsDimensionlessDense    │
│ Matrix()                        │
│ - ComputeStencilInteriorGrid()  │
│ - ComputeRationalBasisNull      │
│ Space()                         │
│ - ComputePreliminaryApproximations() │
│ - ComputeWeights()              │
│ - ComputeStencilBoundaryGrid()  │
│ - AssembleOperator()            │
└─────────────────────────────────┘
```
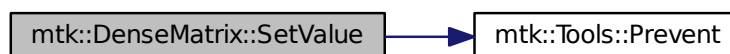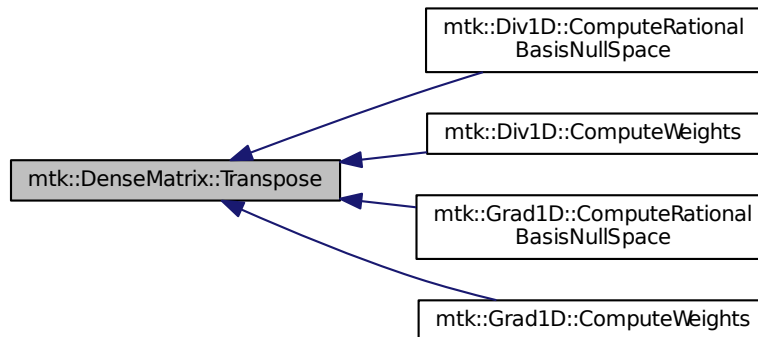
Public Member Functions

- Grad1D ()

    Default constructor.

- Grad1D (const Grad1D &grad)

      Copy constructor.

- ∼Grad1D ()

      Destructor.

- bool ConstructGrad1D (int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimetic↩
  Threshold)

      Factory method implementing the CBS Algorithm to build operator.

- int num_bndy_coeffs () const

      Returns how many coefficients are approximating at the boundary.

- Real ∗ coeffs_interior () const

      Returns coefficients for the interior of the grid.

- Real ∗ weights_crs (void) const

      Returns collection of weights as computed by the CRSA.

- Real ∗ weights_cbs (void) const

      Returns collection of weights as computed by the CBSA.

- DenseMatrix mim_bndy () const

      Return collection of mimetic approximations at the boundary.

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

      Returns the operator as a dense matrix.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid) const

      Returns the operator as a dense matrix.

- DenseMatrix ReturnAsDimensionlessDenseMatrix (int num_cells_x) const

      Returns the operator as a dimensionless dense matrix.

Private Member Functions

- bool ComputeStencilInteriorGrid (void)

      Stage 1 of the CBS Algorithm.

- bool ComputeRationalBasisNullSpace (void)

      Stage 2.1 of the CBS Algorithm.

- bool ComputePreliminaryApproximations (void)

      Stage 2.2 of the CBS Algorithm.

- bool ComputeWeights (void)

      Stage 2.3 of the CBS Algorithm.

- bool ComputeStencilBoundaryGrid (void)

      Stage 2.4 of the CBS Algorithm.

- bool AssembleOperator (void)

      Stage 3 of the CBS Algorithm.

Private Attributes

- int order_accuracy_

      Order of numerical accuracy of the operator.

- int dim_null_

      Dim. null-space for boundary approximations.

- int num_bndy_approxs_

Req. approximations at and near the boundary.

- int num_bndy_coeffs_

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

- int gradient_length_

  Length of the output array.

- int minrow_

  Row from the optimizer with the minimum rel. nor.

- int row_

  Row currently processed by the optimizer.

- DenseMatrix rat_basis_null_space_

  Rational b. null-space w. bndy.

- Real ∗ coeffs_interior_

  Interior stencil.

- Real ∗ prem_apps_

  2D array of boundary preliminary approximations.

- Real ∗ weights_crs_

  Array containing weights from CRSA.

- Real ∗ weights_cbs_

  Array containing weights from CBSA.

- Real ∗ mim_bndy_

  Array containing mimetic boundary approximations.

- Real ∗ gradient_

  Output array containing the operator and weights.

- Real mimetic_threshold_

  $<$ *Mimetic threshold.*

Friends

- std::ostream & operator$<<$ (std::ostream &stream, Grad1D &in)

  Output stream operator for printing.

16.8.1 Detailed Description

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

Definition at line 81 of file mtk_grad_1d.h.

16.8.2 Constructor & Destructor Documentation

16.8.2.1 mtk::Grad1D::Grad1D ( )

Definition at line 129 of file mtk_grad_1d.cc.

16.8.2.2 mtk::Grad1D::Grad1D (const Grad1D & *grad*)

Parameters

in		*div*	Given divergence.

Definition at line 145 of file mtk_grad_1d.cc.

16.8.2.3 mtk::Grad1D::∼Grad1D ( )

Definition at line 161 of file mtk_grad_1d.cc.

16.8.3 Member Function Documentation

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

Construct the output array with the operator and its weights.

1. The first entry of the array will contain the order of accuracy.

2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.

3. The third entry will contain the collection of weights.

4. The next dim_null + 1 entries will contain the collections of approximating coefficients for the west boundary of the grid.

Definition at line 1499 of file mtk_grad_1d.cc.

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

Returns

   Coefficients for the interior of the grid.

Definition at line 326 of file mtk_grad_1d.cc.

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

Compute the set of preliminary approximations on the boundary neighborhood.

1. Create generator vector for the first approximation.

2. Compute the dim_null near-the-boundary columns of the pi matrix.

3. Create the Vandermonde matrix for this iteration.

4. New order-selector vector (gets re-written with LAPACK solutions).

5. Solving TT∗rr = ob yields the columns rr of the kk matrix.

6. Scale the kk matrix to make it a rational basis for null-space.

7. Extract the last dim_null values of the pre-scaled ob.

8. Once we posses the bottom elements, we proceed with the scaling.

Definition at line 831 of file mtk_grad_1d.cc.

Here is the call graph for this function:
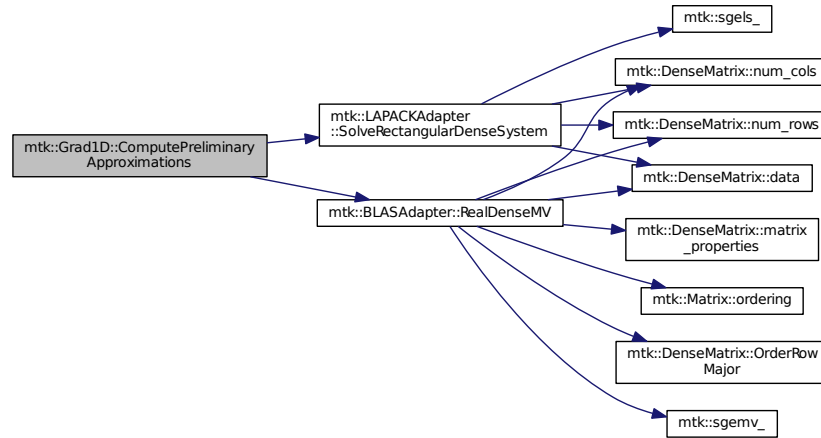
16.8.3.4 bool mtk::Grad1D::ComputeRationalBasisNullSpace (void) `[private]`

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

1. Create generator vector for the first approximation.

2. Create Vandermonde matrix.

3. QR-factorize the Vandermonde matrix.

4. Extract the basis for the null-space from Q matrix.

5. Scale null-space to make it rational.

Definition at line 648 of file mtk_grad_1d.cc.

Here is the call graph for this function:
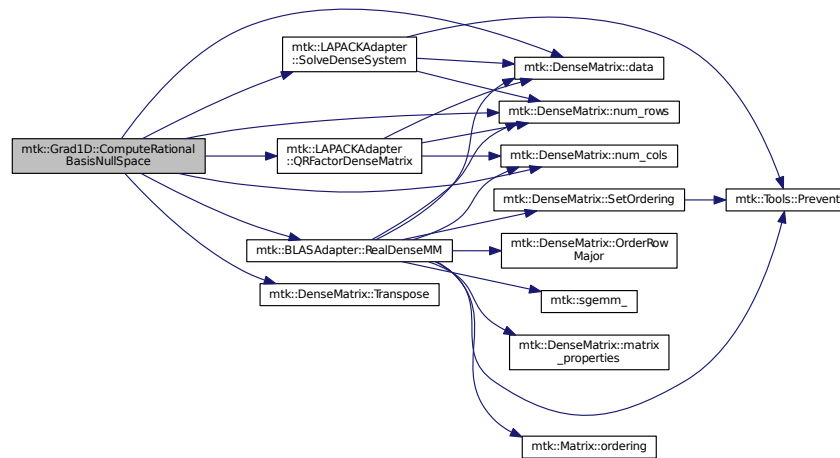
16.8.3.5 bool mtk::Grad1D::ComputeStencilBoundaryGrid (void) `[private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.

2. Compute alpha values.

3. Compute the mimetic boundary approximations.

Definition at line 1393 of file mtk_grad_1d.cc.

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

Compute the stencil approximating the interior of the staggered grid.

1. Create vector for interior spatial coordinates.

2. Create Vandermonde matrix (using interior coordinates as generator).

3. Create order-selector vector.

4. Solve dense Vandermonde system to attain the interior coefficients.

Definition at line 551 of file mtk_grad_1d.cc.

Here is the call graph for this function:

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

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

1. Construct the ■ matrix.

2. Use interior stencil to build proper RHS vector **h**.

3. Get weights (as **CRSA**): ■**q** = **h**.

4. If required order is greater than critical order, start the **CBSA**.

5. Create ■ matrix from ■.

6. Prepare constraint vector as in the CBSA: ■.

7. Brute force search through all the rows of the Φ matrix.

8. Apply solution found from brute force search.

Definition at line 1052 of file mtk_grad_1d.cc.

Here is the call graph for this function:
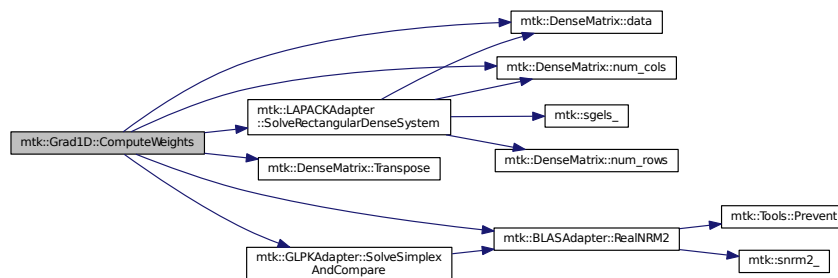
16.8.3.8 bool mtk::Grad1D::ConstructGrad1D (int *order_accuracy* = kDefaultOrderAccuracy, Real *mimetic_threshold* = kDefaultMimeticThreshold)
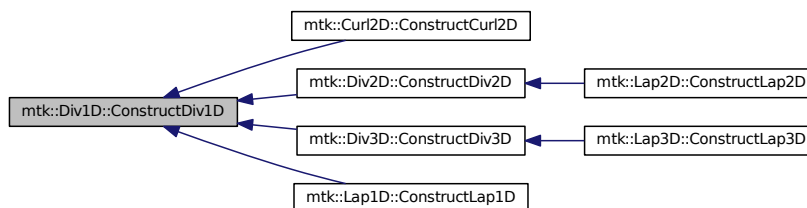
Returns

Success of the solution.

1. Compute stencil for the interior cells.

2. Compute a rational null-space from the first matrix transposed.

3. Compute preliminary approximation (non-mimetic) on the boundaries.

4. Compute quadrature weights to impose the mimetic conditions.

5. Compute real approximation (mimetic) on the boundaries.

6. Assemble operator.

Definition at line 182 of file mtk_grad_1d.cc.

Here is the call graph for this function:

Here is the caller graph for this function:
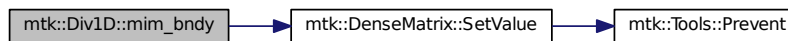

16.8.3.9   mtk::DenseMatrix mtk::Grad1D::mim_bndy ( ) const

Returns

Collection of mimetic approximations at the boundary.

Definition at line 341 of file mtk_grad_1d.cc.

Here is the call graph for this function:

```
┌─────────────────────────┐     ┌───────────────────────────┐     ┌──────────────────────┐
│ mtk::Grad1D::mim_bndy   │ ──▶ │ mtk::DenseMatrix::SetValue │ ──▶ │ mtk::Tools::Prevent  │
└─────────────────────────┘     └───────────────────────────┘     └──────────────────────┘
```

Here is the caller graph for this function:

```
┌─────────────────────────┐     ┌───────────────────────────┐
│ mtk::Grad1D::mim_bndy   │ ◀── │ mtk::RobinBCDescriptor1D  │
└─────────────────────────┘     │ ::ImposeOnLaplacianMatrix │
                                └───────────────────────────┘
```

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

Returns

How many coefficients are approximating at the boundary.
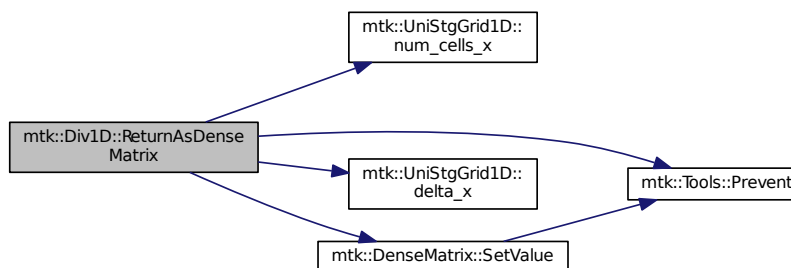
Definition at line 321 of file mtk_grad_1d.cc.

16.8.3.11 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (mtk::Real *west,* mtk::Real *east,* int *num_cells_x*) const
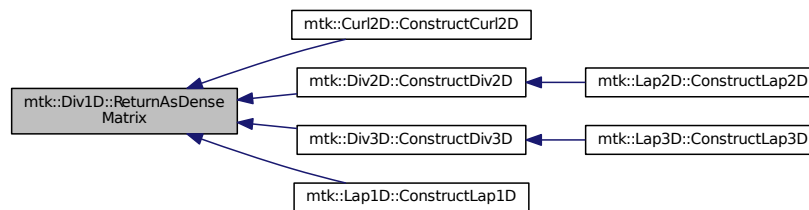
Returns

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 356 of file mtk_grad_1d.cc.

Here is the call graph for this function:
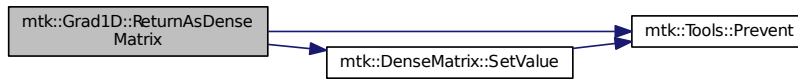
Here is the caller graph for this function:
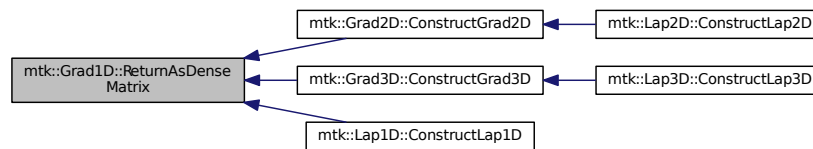
16.8.3.12 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (const UniStgGrid1D & *grid*) const

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 425 of file mtk_grad_1d.cc.

Here is the call graph for this function:
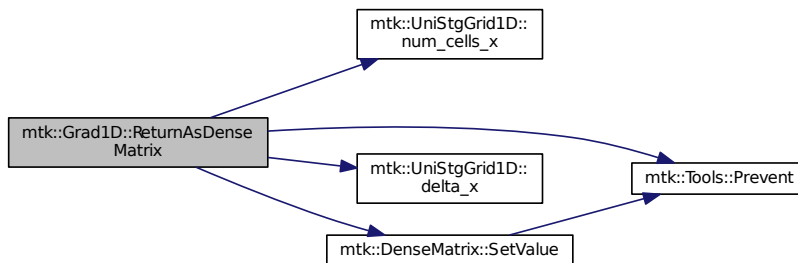
16.8.3.13 **mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix (int *num_cells_x*) const**

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 489 of file mtk_grad_1d.cc.

Here is the call graph for this function:

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

Returns

Collection of weights as computed by the CBSA.

Definition at line 336 of file mtk_grad_1d.cc.

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

Returns

Success of the solution.

Definition at line 331 of file mtk_grad_1d.cc.

16.8.4 Friends And Related Function Documentation

16.8.4.1 **std::ostream& operator≪ (std::ostream & *stream,* mtk::Grad1D & *in*)** `[friend]`

1. Print order of accuracy.

2. Print approximating coefficients for the interior.

3. Print mimetic weights.

4. Print mimetic approximations at the boundary.

Definition at line 79 of file mtk_grad_1d.cc.

16.8.5 Member Data Documentation

16.8.5.1 Real∗ mtk::Grad1D::coeffs_interior_ [private]

Definition at line 217 of file mtk_grad_1d.h.

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

Definition at line 208 of file mtk_grad_1d.h.

16.8.5.3 Real∗ mtk::Grad1D::gradient_ [private]

Definition at line 222 of file mtk_grad_1d.h.

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

Definition at line 211 of file mtk_grad_1d.h.

16.8.5.5 Real∗ mtk::Grad1D::mim_bndy_ [private]

Definition at line 221 of file mtk_grad_1d.h.

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

Definition at line 224 of file mtk_grad_1d.h.

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

Definition at line 212 of file mtk_grad_1d.h.

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

Definition at line 209 of file mtk_grad_1d.h.

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

Definition at line 210 of file mtk_grad_1d.h.

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

Definition at line 207 of file mtk_grad_1d.h.

16.8.5.11 Real∗ mtk::Grad1D::prem_apps_ [private]

Definition at line 218 of file mtk_grad_1d.h.

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

Definition at line 215 of file mtk_grad_1d.h.

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

Definition at line 213 of file mtk_grad_1d.h.

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

Definition at line 220 of file mtk_grad_1d.h.

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

Definition at line 219 of file mtk_grad_1d.h.
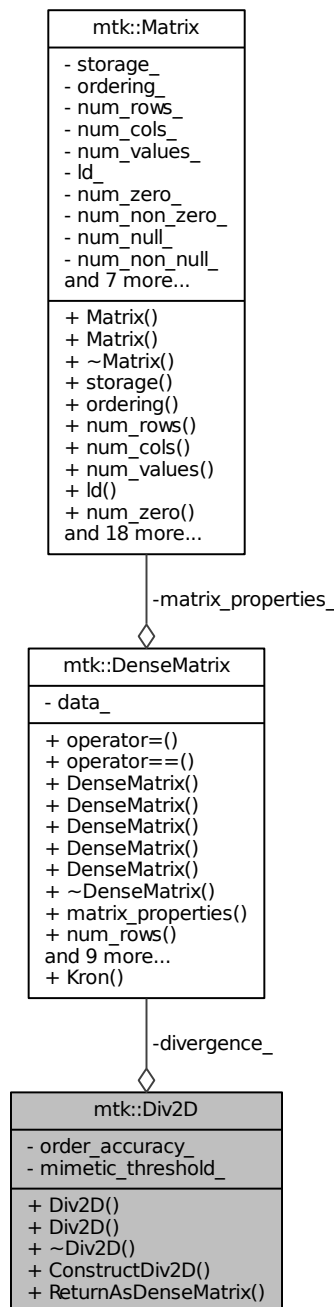
The documentation for this class was generated from the following files:

- include/mtk_grad_1d.h

- src/mtk_grad_1d.cc

16.9 mtk::Grad2D Class Reference

Implements a 2D mimetic gradient operator.

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

Collaboration diagram for mtk::Grad2D:

```
┌─────────────────────────┐
│      mtk::Matrix        │
├─────────────────────────┤
│ - storage_              │
│ - ordering_             │
│ - num_rows_             │
│ - num_cols_             │
│ - num_values_           │
│ - ld_                   │
│ - num_zero_             │
│ - num_non_zero_         │
│ - num_null_             │
│ - num_non_null_         │
│ and 7 more...           │
├─────────────────────────┤
│ + Matrix()              │
│ + Matrix()              │
│ + ~Matrix()             │
│ + storage()             │
│ + ordering()            │
│ + num_rows()            │
│ + num_cols()            │
│ + num_values()          │
│ + ld()                  │
│ + num_zero()            │
│ and 18 more...          │
└─────────────────────────┘
             │ -matrix_properties_
             ◇
┌─────────────────────────┐
│    mtk::DenseMatrix     │
├─────────────────────────┤
│ - data_                 │
├─────────────────────────┤
│ + operator=()           │
│ + operator==()          │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ and 9 more...           │
│ + Kron()                │
└─────────────────────────┘
             │ -gradient_
             ◇
┌─────────────────────────┐
│      mtk::Grad2D        │
├─────────────────────────┤
│ - order_accuracy_       │
│ - mimetic_threshold_    │
├─────────────────────────┤
│ + Grad2D()              │
│ + Grad2D()              │
│ + ~Grad2D()             │
│ + ConstructGrad2D()     │
│ + ReturnAsDenseMatrix() │
└─────────────────────────┘
```

Public Member Functions

- Grad2D ()

Default constructor.

- Grad2D (const Grad2D &grad)

    Copy constructor.

- ∼Grad2D ()

    Destructor.
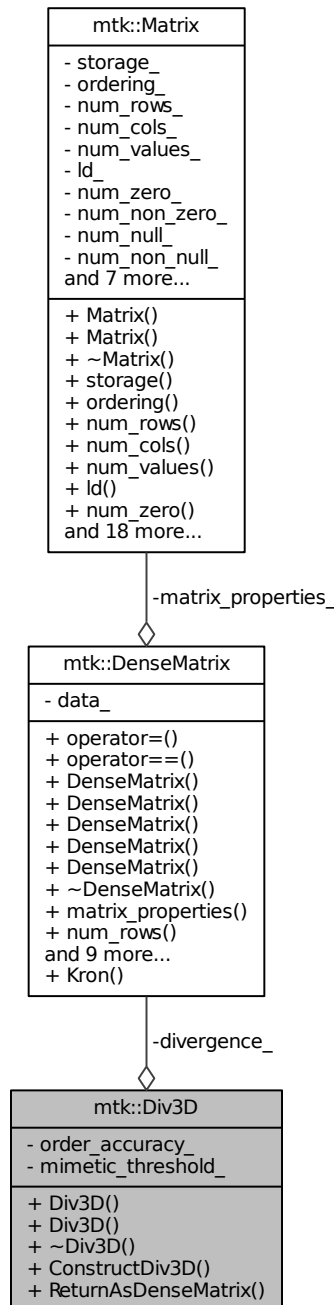
- bool ConstructGrad2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_↩
    threshold=kDefaultMimeticThreshold)

    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.

16.9.1 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.

16.9.2 Constructor & Destructor Documentation

16.9.2.1 mtk::Grad2D::Grad2D ( )

Definition at line 67 of file mtk_grad_2d.cc.

16.9.2.2 mtk::Grad2D::Grad2D (const Grad2D & *grad*)

Parameters

in		*div*	Given divergence.

Definition at line 71 of file mtk_grad_2d.cc.

16.9.2.3 mtk::Grad2D::∼Grad2D ( )

Definition at line 75 of file mtk_grad_2d.cc.

16.9.3 Member Function Documentation

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

Returns

Success of the construction.

Definition at line 77 of file mtk_grad_2d.cc.

Here is the call graph for this function:
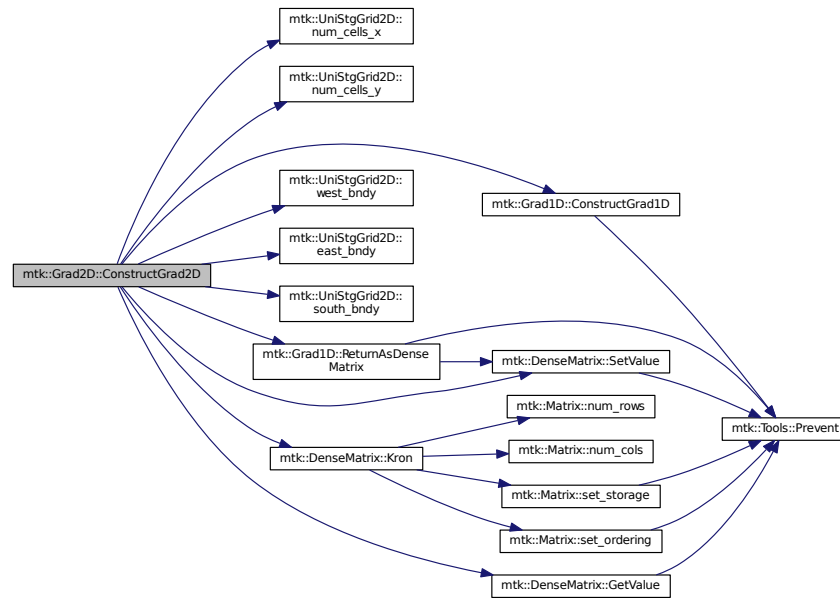
Here is the caller graph for this function:

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

16.9.4 Member Data Documentation

16.9.4.1 DenseMatrix mtk::Grad2D::gradient_ `[private]`

Definition at line 108 of file mtk_grad_2d.h.

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

Definition at line 112 of file mtk_grad_2d.h.

16.9.4.3 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

16.10 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.

16.10.1 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.

16.10.2 Constructor & Destructor Documentation

16.10.2.1 mtk::Grad3D::Grad3D ( )

Definition at line 67 of file mtk_grad_3d.cc.

16.10.2.2 mtk::Grad3D::Grad3D (const Grad3D & *grad*)

Parameters

in	*div*	Given divergence.

Definition at line 71 of file mtk_grad_3d.cc.

16.10.2.3 mtk::Grad3D::∼Grad3D ( )

Definition at line 75 of file mtk_grad_3d.cc.

16.10.3 Member Function Documentation

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

Returns

Success of the construction.

1. Build preliminary staggering through the x direction.

2. Build preliminary staggering through the y direction.

3. Build preliminary staggering through the z direction.

4. Actual operator: GG_xyz = [gx; gy; gz].

Definition at line 77 of file mtk_grad_3d.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

16.10.3.2 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:

16.10.4 Member Data Documentation

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

Definition at line 108 of file mtk_grad_3d.h.

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

Definition at line 112 of file mtk_grad_3d.h.

16.10.4.3 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

16.11 mtk::Interp1D Class Reference

Implements a 1D interpolation operator.

`#include <mtk_interp_1d.h>`

Collaboration diagram for mtk::Interp1D:

```
┌─────────────────────────────┐
│        mtk::Interp1D        │
├─────────────────────────────┤
│ - dir_interp_               │
│ - order_accuracy_           │
│ - coeffs_interior_          │
├─────────────────────────────┤
│ + Interp1D()                │
│ + Interp1D()                │
│ + ~Interp1D()               │
│ + ConstructInterp1D()       │
│ + coeffs_interior()         │
│ + ReturnAsDenseMatrix()     │
└─────────────────────────────┘
```

Public Member Functions

- Interp1D ()

    Default constructor.
- Interp1D (const Interp1D &interp)

    Copy constructor.
- ~Interp1D ()

    Destructor.
- bool ConstructInterp1D (int order_accuracy=kDefaultOrderAccuracy, mtk::DirInterp dir=SCALAR_TO_VECTOR)

    Factory method to build operator.
- Real ∗ coeffs_interior () const

    Returns coefficients for the interior of the grid.
- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid) 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.

16.11.1 Detailed Description

This class implements a 1D interpolation operator.

Definition at line 82 of file mtk_interp_1d.h.

16.11.2 Constructor & Destructor Documentation

16.11.2.1 mtk::Interp1D::Interp1D ( )

Definition at line 80 of file mtk_interp_1d.cc.

16.11.2.2 mtk::Interp1D::Interp1D (const **Interp1D** & *interp*)

Parameters

in	*interp*	Given interpolation operator.

Definition at line 85 of file mtk_interp_1d.cc.

16.11.2.3 mtk::Interp1D::∼Interp1D ( )

Definition at line 90 of file mtk_interp_1d.cc.

16.11.3 Member Function Documentation

16.11.3.1 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.

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

Returns

Success of the solution.

1. Compute stencil for the interior cells.

Definition at line 96 of file mtk_interp_1d.cc.

Here is the call graph for this function:
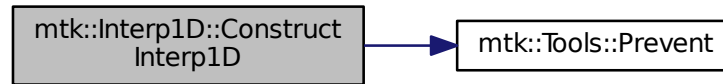
16.11.3.3 mtk::DenseMatrix mtk::Interp1D::ReturnAsDenseMatrix (const UniStgGrid1D & *grid*) const

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:
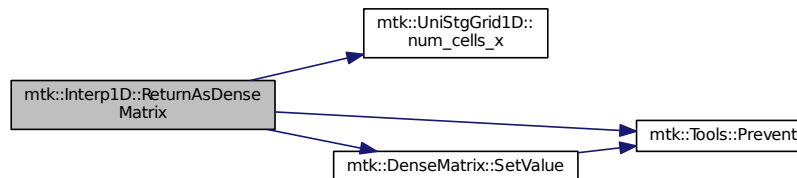
16.11.4 Friends And Related Function Documentation

16.11.4.1 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.

16.11.5 Member Data Documentation

16.11.5.1 Real∗ mtk::Interp1D::coeffs_interior_ `[private]`

Definition at line 127 of file mtk_interp_1d.h.

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

Definition at line 123 of file mtk_interp_1d.h.

16.11.5.3 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

16.12 mtk::Interp2D Class Reference

Implements a 2D interpolation operator.

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

Collaboration diagram for mtk::Interp2D:
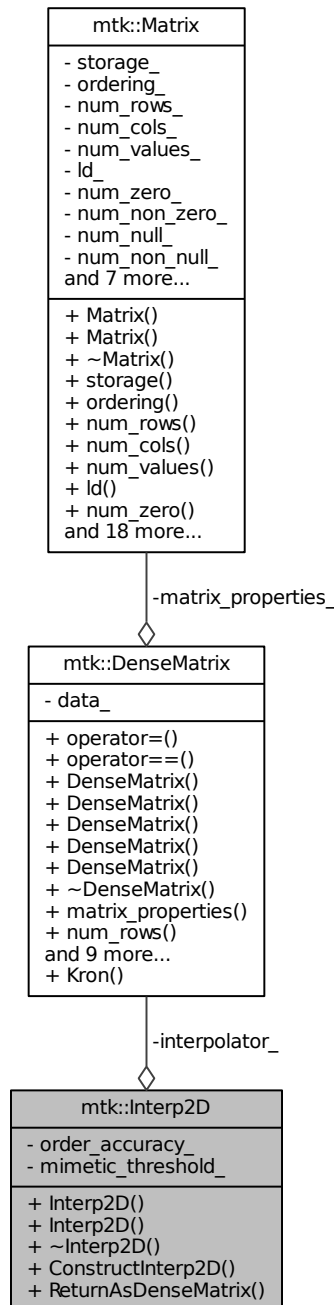
Public Member Functions

- 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.

16.12.1 Detailed Description

This class implements a 2D interpolation operator.

Definition at line 76 of file mtk_interp_2d.h.

16.12.2 Constructor & Destructor Documentation

16.12.2.1 mtk::Interp2D::Interp2D ( )

16.12.2.2 mtk::Interp2D::Interp2D (const Interp2D & *interp*)

Parameters

in	*lap*	Given Laplacian.

16.12.2.3 mtk::Interp2D::∼Interp2D ( )

16.12.3 Member Function Documentation

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

Returns

Success of the construction.

16.12.3.2 DenseMatrix mtk::Interp2D::ReturnAsDenseMatrix ( )

Returns

The operator as a dense matrix.

16.12.4 Member Data Documentation

16.12.4.1 DenseMatrix mtk::Interp2D::interpolator_ `[private]`

Definition at line 108 of file mtk_interp_2d.h.

16.12.4.2 Real mtk::Interp2D::mimetic_threshold_ `[private]`

Definition at line 112 of file mtk_interp_2d.h.

16.12.4.3 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

16.13 mtk::Lap1D Class Reference

Implements a 1D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap1D:

```
┌─────────────────────────────────┐
│           mtk::Lap1D            │
├─────────────────────────────────┤
│ - order_accuracy_               │
│ - laplacian_length_             │
│ - laplacian_                    │
│ - delta_                        │
│ - mimetic_threshold_            │
├─────────────────────────────────┤
│ + Lap1D()                       │
│ + Lap1D()                       │
│ + ~Lap1D()                      │
│ + order_accuracy()              │
│ + mimetic_threshold()           │
│ + delta()                       │
│ + ConstructLap1D()              │
│ + ReturnAsDenseMatrix()         │
│ + data()                        │
└─────────────────────────────────┘
```

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=kDefaultMimetic↩
  Threshold)

    Factory method implementing the CBS Algorithm to build operator.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid) const

    Return the operator as a dense matrix.

- const mtk::Real ∗ data (const UniStgGrid1D &grid) const

    Return the operator as a dense array.

Private Attributes

- int order_accuracy_

  Order of numerical accuracy of the operator.

- int laplacian_length_

  Length of the output array.

- Real ∗ laplacian_

  Output array containing the operator and weights.

- Real delta_

  $<$ *If 0.0, then this Laplacian is dimensionless.*

- Real mimetic_threshold_

  $<$ *Mimetic threshold.*

Friends

- std::ostream & operator$<<$ (std::ostream &stream, Lap1D &in)

  Output stream operator for printing.

16.13.1   Detailed Description

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

Definition at line 76 of file mtk_lap_1d.h.

16.13.2   Constructor & Destructor Documentation

16.13.2.1   mtk::Lap1D::Lap1D ( )

Definition at line 108 of file mtk_lap_1d.cc.

16.13.2.2   mtk::Lap1D::Lap1D (const Lap1D & *lap*)

Parameters

in	*lap*	Given Laplacian.

16.13.2.3   mtk::Lap1D::∼Lap1D ( )

Definition at line 114 of file mtk_lap_1d.cc.

16.13.3   Member Function Documentation

16.13.3.1   bool mtk::Lap1D::ConstructLap1D (int *order_accuracy* = kDefaultOrderAccuracy, mtk::Real *mimetic_threshold* = kDefaultMimeticThreshold)
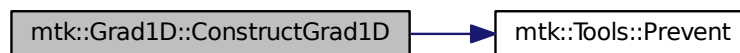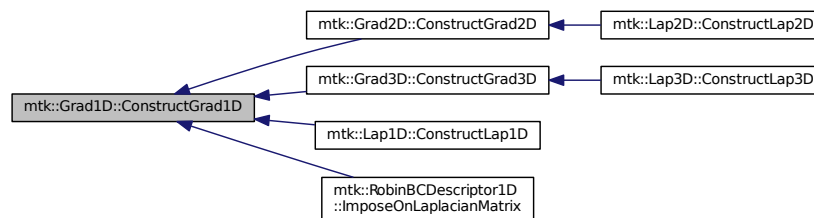
Returns

> 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: $\breve{\mathbf{L}}_x^k = \breve{\mathbf{D}}_x^k \breve{\mathbf{G}}_x^k$

5. Extract the coefficients from the matrix and store them in the array.

Warning

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

1. The first entry of the array will contain the order of accuracy.

2. The second entry of the array will contain the collection of approximating coefficients for the interior of the grid.

3. We DO NOT have weights in this operator. Copy mimetic bndy coeffs.

Definition at line 135 of file mtk_lap_1d.cc.

Here is the call graph for this function:
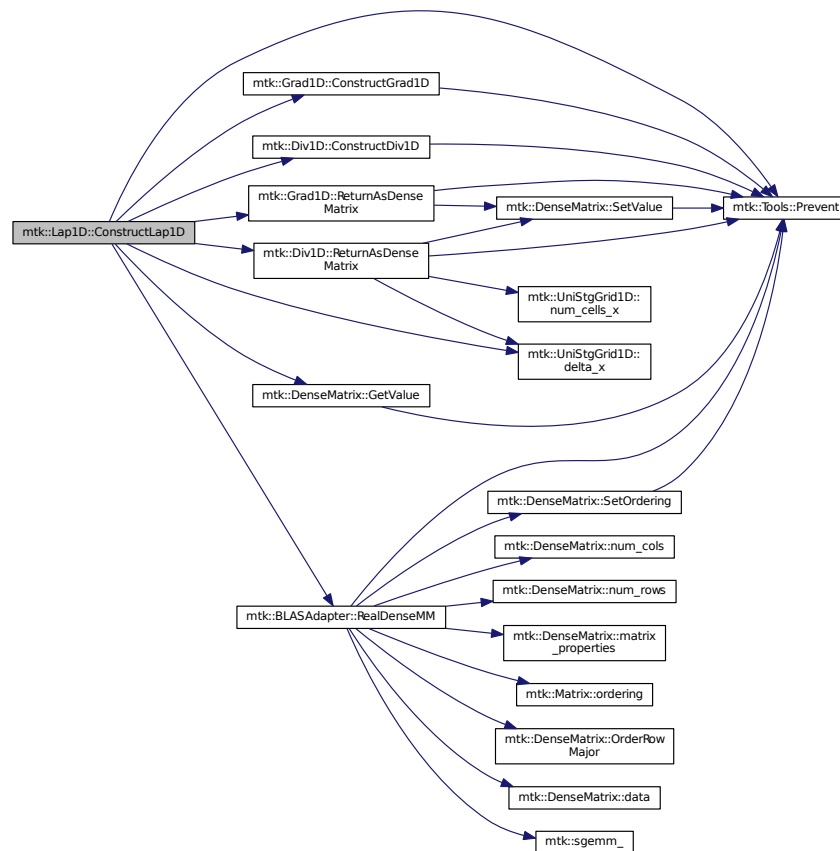
16.13.3.2   const mtk::Real $*$ **mtk::Lap1D::data (const UniStgGrid1D &** *grid* **) const**

Returns

The operator as a dense array.

Definition at line 356 of file mtk_lap_1d.cc.

Here is the call graph for this function:

```
┌─────────────────────┐       ┌─────────────────────────┐
│  mtk::Lap1D::data   │──────▶│  mtk::DenseMatrix::data │
└─────────────────────┘       └─────────────────────────┘
```
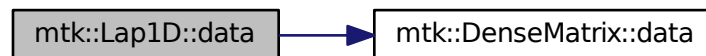
16.13.3.3   mtk::Real mtk::Lap1D::delta ( ) const

Returns

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

Definition at line 130 of file mtk_lap_1d.cc.

Here is the caller graph for this function:

```
┌─────────────────────┐       ┌──────────────────────────────┐
│  mtk::Lap1D::delta  │◀──────│  mtk::RobinBCDescriptor1D    │
└─────────────────────┘       │  ::ImposeOnLaplacianMatrix   │
                              └──────────────────────────────┘
```

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

Returns

>   Mimetic threshold used in the CBS algorithm to construct operator.

Definition at line 125 of file mtk_lap_1d.cc.

Here is the caller graph for this function:

```
┌─────────────────┐      ┌──────────────────────────┐
│ mtk::Lap1D::mimetic │ ◄──── │ mtk::RobinBCDescriptor1D │
│    _threshold    │      │ ::ImposeOnLaplacianMatrix │
└─────────────────┘      └──────────────────────────┘
```
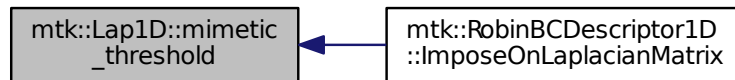
16.13.3.5   int mtk::Lap1D::order_accuracy ( ) const

Returns

>   Order of accuracy of the operator.

Definition at line 120 of file mtk_lap_1d.cc.

Here is the caller graph for this function:

```
┌────────────────────────┐      ┌──────────────────────────┐
│ mtk::Lap1D::order_accuracy │ ◄──── │ mtk::RobinBCDescriptor1D │
└────────────────────────┘      │ ::ImposeOnLaplacianMatrix │
                                 └──────────────────────────┘
```
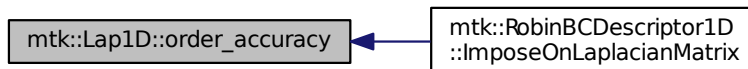
16.13.3.6   mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix (const UniStgGrid1D & *grid*) const

Returns

>   The operator as a dense matrix.

1. Extract mimetic coefficients from the west boundary.

2. Extract interior coefficients.

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

Note

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

Definition at line 286 of file mtk_lap_1d.cc.

Here is the call graph for this function:
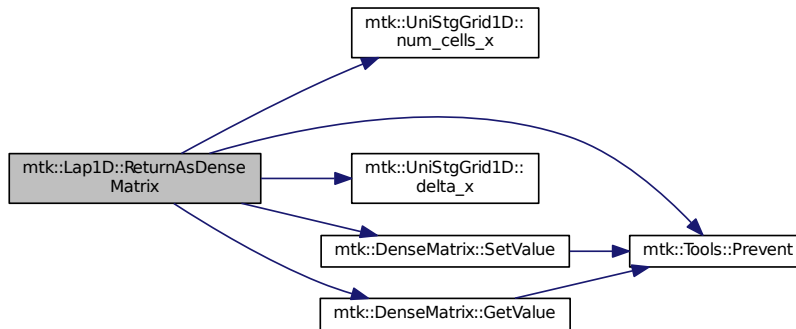
16.13.4 Friends And Related Function Documentation

16.13.4.1 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.

16.13.5 Member Data Documentation

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

Definition at line 143 of file mtk_lap_1d.h.

16.13.5.2 Real∗ mtk::Lap1D::laplacian_ `[private]`

Definition at line 141 of file mtk_lap_1d.h.

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

Definition at line 139 of file mtk_lap_1d.h.

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

Definition at line 145 of file mtk_lap_1d.h.

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

Definition at line 138 of file mtk_lap_1d.h.

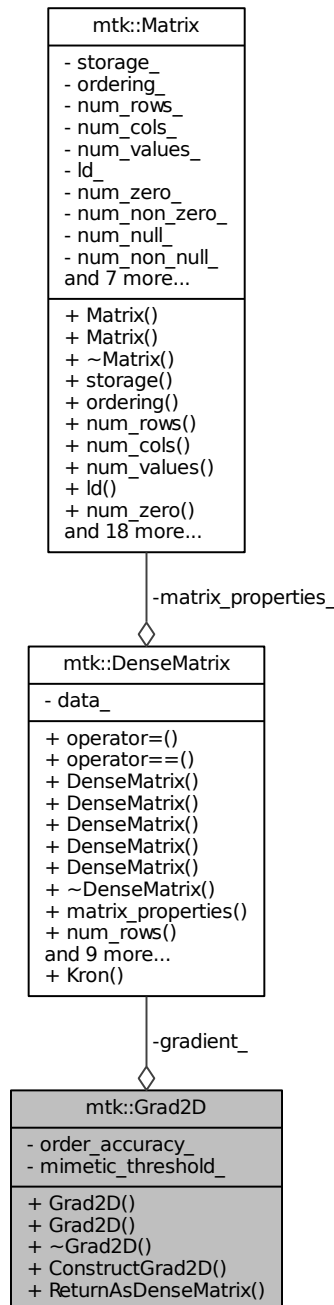The documentation for this class was generated from the following files:

- include/mtk_lap_1d.h

- src/mtk_lap_1d.cc

16.14   mtk::Lap2D Class Reference

Implements a 2D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap2D:

```
                    ┌─────────────────────────┐
                    │       mtk::Matrix       │
                    ├─────────────────────────┤
                    │ - storage_              │
                    │ - ordering_             │
                    │ - num_rows_             │
                    │ - num_cols_             │
                    │ - num_values_           │
                    │ - ld_                   │
                    │ - num_zero_             │
                    │ - num_non_zero_         │
                    │ - num_null_             │
                    │ - num_non_null_         │
                    │ and 7 more...           │
                    ├─────────────────────────┤
                    │ + Matrix()              │
                    │ + Matrix()              │
                    │ + ~Matrix()             │
                    │ + storage()             │
                    │ + ordering()            │
                    │ + num_rows()            │
                    │ + num_cols()            │
                    │ + num_values()          │
                    │ + ld()                  │
                    │ + num_zero()            │
                    │ and 18 more...          │
                    └─────────────────────────┘
                                 │
                                 │ -matrix_properties_
                                 ◇
                    ┌─────────────────────────┐
                    │    mtk::DenseMatrix      │
                    ├─────────────────────────┤
                    │ - data_                 │
                    ├─────────────────────────┤
                    │ + operator=()           │
                    │ + operator==()          │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + ~DenseMatrix()        │
                    │ + matrix_properties()   │
                    │ + num_rows()            │
                    │ and 9 more...           │
                    │ + Kron()                │
                    └─────────────────────────┘
                                 │
                                 │ -laplacian_
                                 ◇
                    ┌─────────────────────────┐
                    │       mtk::Lap2D        │
                    ├─────────────────────────┤
                    │ - order_accuracy_       │
                    │ - mimetic_threshold_    │
                    ├─────────────────────────┤
                    │ + Lap2D()               │
                    │ + Lap2D()               │
                    │ + ~Lap2D()              │
                    │ + ConstructLap2D()      │
                    │ + ReturnAsDenseMatrix() │
                    │ + data()                │
                    └─────────────────────────┘
```

Public Member Functions

- Lap2D ()

Default constructor.

• Lap2D (const Lap2D &lap)

Copy constructor.

• ∼Lap2D ()

Destructor.
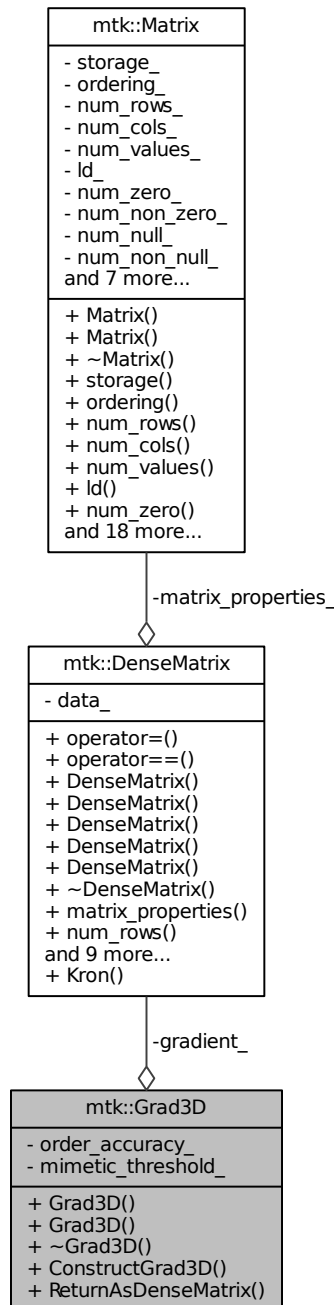
• bool ConstructLap2D (const UniStgGrid2D &grid, int order_accuracy=kDefaultOrderAccuracy, Real mimetic_↩
threshold=kDefaultMimeticThreshold)

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.

16.14.1 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.

16.14.2 Constructor & Destructor Documentation

16.14.2.1 mtk::Lap2D::Lap2D ( )

Definition at line 69 of file mtk_lap_2d.cc.

16.14.2.2 mtk::Lap2D::Lap2D (const Lap2D & *lap*)

Parameters

in	*lap*	Given Laplacian.

Definition at line 71 of file mtk_lap_2d.cc.

16.14.2.3 mtk::Lap2D::∼Lap2D ( )

Definition at line 75 of file mtk_lap_2d.cc.

16.14.3 Member Function Documentation

16.14.3.1 bool mtk::Lap2D::ConstructLap2D (const **UniStgGrid2D** & *grid,* int *order_accuracy =* **kDefaultOrderAccuracy,** mtk::Real *mimetic_threshold =* **kDefaultMimeticThreshold**)
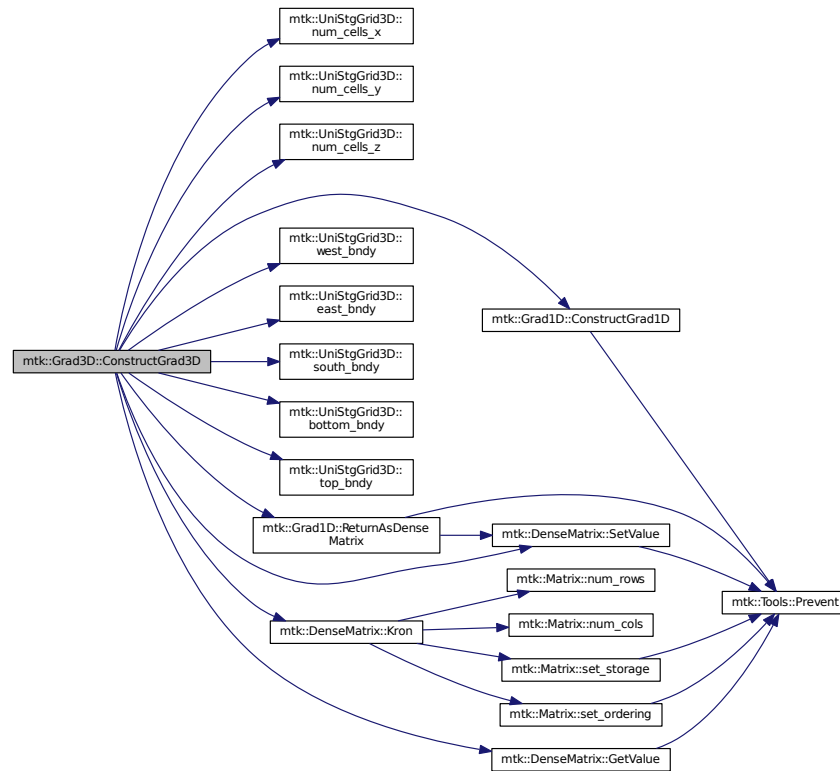
Returns

Success of the construction.

Definition at line 77 of file mtk_lap_2d.cc.

Here is the call graph for this function:
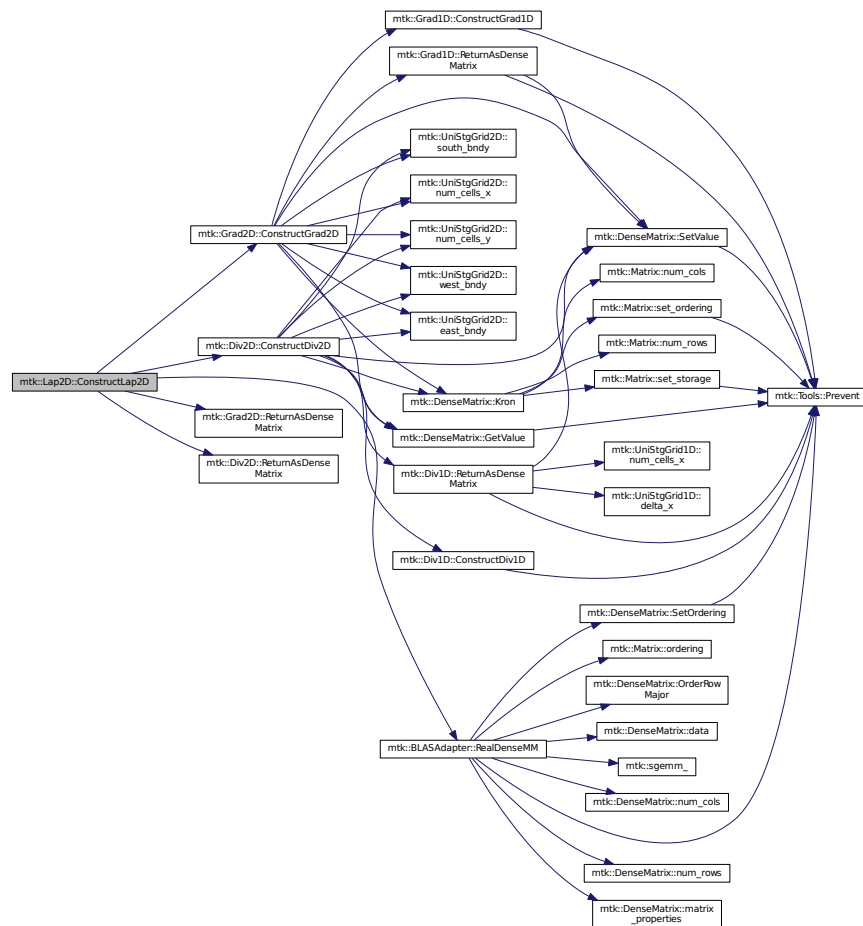
16.14.3.2 **mtk::Real** ∗ **mtk::Lap2D::data (** **) const**

Returns

The operator as a dense array.

Definition at line 115 of file mtk_lap_2d.cc.

16.14.3.3 mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix (   ) const

Returns

The operator as a dense matrix.

Definition at line 110 of file mtk_lap_2d.cc.

16.14.4 Member Data Documentation

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

Definition at line 115 of file mtk_lap_2d.h.

16.14.4.2 Real mtk::Lap2D::mimetic_threshold_ `[private]`

Definition at line 119 of file mtk_lap_2d.h.

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

Definition at line 117 of file mtk_lap_2d.h.

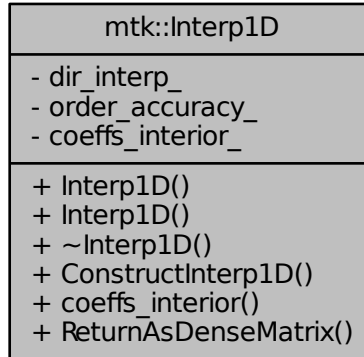The documentation for this class was generated from the following files:

- include/mtk_lap_2d.h

- src/mtk_lap_2d.cc

16.15   mtk::Lap3D Class Reference

Implements a 3D mimetic Laplacian operator.

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

Collaboration diagram for mtk::Lap3D:

```
┌─────────────────────────┐
│      mtk::Matrix        │
├─────────────────────────┤
│ - storage_              │
│ - ordering_             │
│ - num_rows_             │
│ - num_cols_             │
│ - num_values_           │
│ - ld_                   │
│ - num_zero_             │
│ - num_non_zero_         │
│ - num_null_             │
│ - num_non_null_         │
│ and 7 more...           │
├─────────────────────────┤
│ + Matrix()              │
│ + Matrix()              │
│ + ~Matrix()             │
│ + storage()             │
│ + ordering()            │
│ + num_rows()            │
│ + num_cols()            │
│ + num_values()          │
│ + ld()                  │
│ + num_zero()            │
│ and 18 more...          │
└─────────────────────────┘
          │ -matrix_properties_
          ◇
┌─────────────────────────┐
│   mtk::DenseMatrix      │
├─────────────────────────┤
│ - data_                 │
├─────────────────────────┤
│ + operator=()           │
│ + operator==()          │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ and 9 more...           │
│ + Kron()                │
└─────────────────────────┘
          │ -laplacian_
          ◇
┌─────────────────────────┐
│      mtk::Lap3D         │
├─────────────────────────┤
│ - order_accuracy_       │
│ - mimetic_threshold_    │
├─────────────────────────┤
│ + operator*()           │
│ + Lap3D()               │
│ + Lap3D()               │
│ + ~Lap3D()              │
│ + ConstructLap3D()      │
│ + ReturnAsDenseMatrix() │
│ + data()                │
└─────────────────────────┘
```

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.

16.15.1 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.

16.15.2 Constructor & Destructor Documentation

16.15.2.1 mtk::Lap3D::Lap3D ( )

Definition at line 76 of file mtk_lap_3d.cc.

16.15.2.2 mtk::Lap3D::Lap3D (const Lap3D & *lap*)

Parameters

in	*lap*	Given Laplacian.

Definition at line 78 of file mtk_lap_3d.cc.

16.15.2.3 mtk::Lap3D::∼Lap3D ( )

Definition at line 82 of file mtk_lap_3d.cc.

16.15.3 Member Function Documentation

16.15.3.1 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:
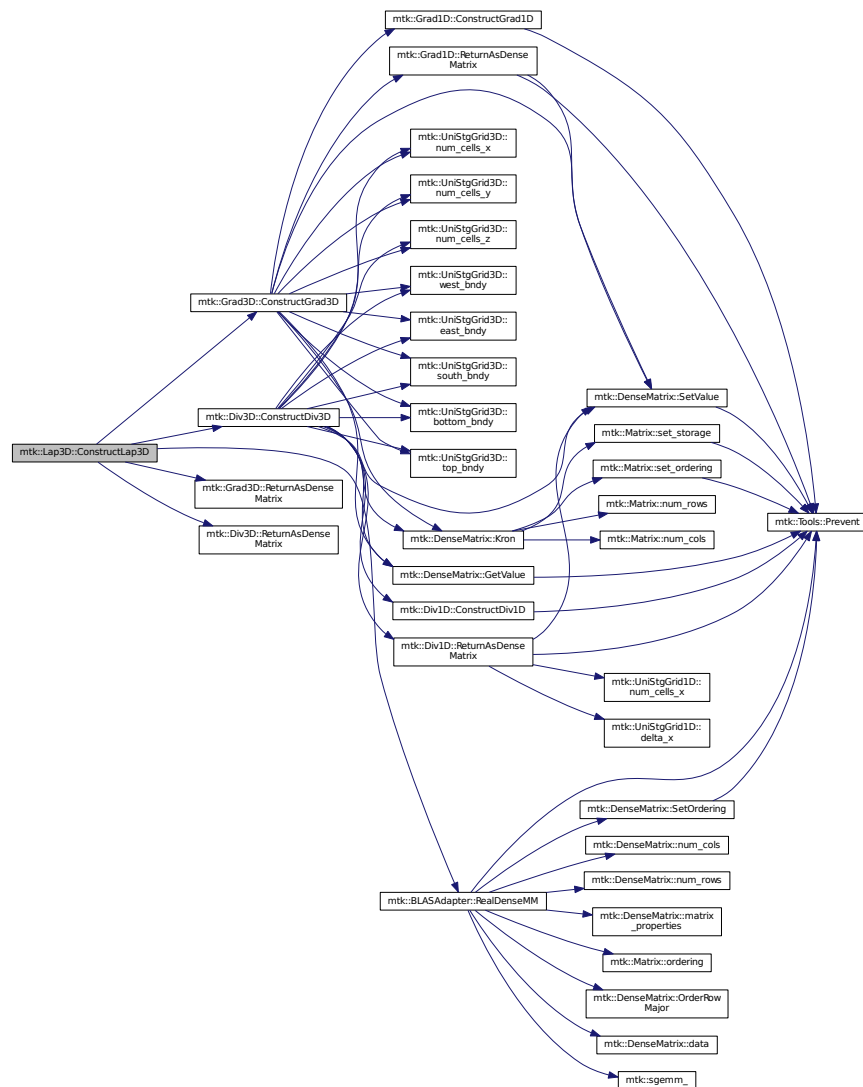
16.15.3.2 mtk::Real ∗ mtk::Lap3D::data ( ) const

Returns

The operator as a dense array.

Definition at line 122 of file mtk_lap_3d.cc.

16.15.3.3 mtk::UniStgGrid3D mtk::Lap3D::operator∗ (const UniStgGrid3D & *grid*) const

Definition at line 69 of file mtk_lap_3d.cc.

16.15.3.4 mtk::DenseMatrix mtk::Lap3D::ReturnAsDenseMatrix ( ) const

Returns

The operator as a dense matrix.

Definition at line 117 of file mtk_lap_3d.cc.

16.15.4 Member Data Documentation

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

Definition at line 118 of file mtk_lap_3d.h.

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

Definition at line 122 of file mtk_lap_3d.h.

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

Definition at line 120 of file mtk_lap_3d.h.

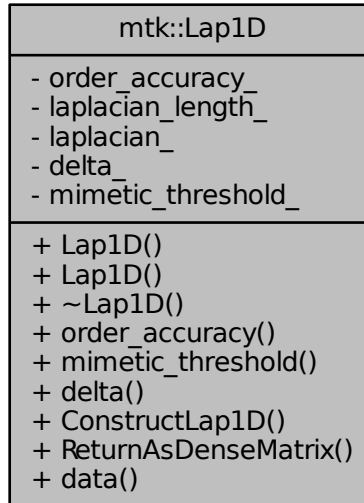The documentation for this class was generated from the following files:

- include/mtk_lap_3d.h
- src/mtk_lap_3d.cc

16.16 mtk::LAPACKAdapter Class Reference

Adapter class for the LAPACK API.

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

Collaboration diagram for mtk::LAPACKAdapter:

```
┌─────────────────────────────────────────────┐
│            mtk::LAPACKAdapter                │
├─────────────────────────────────────────────┤
│                                             │
├─────────────────────────────────────────────┤
│  + SolveDenseSystem()                       │
│  + SolveDenseSystem()                       │
│  + SolveDenseSystem()                       │
│  + SolveDenseSystem()                       │
│  + SolveRectangularDenseSystem()            │
│  + QRFactorDenseMatrix()                    │
└─────────────────────────────────────────────┘
```
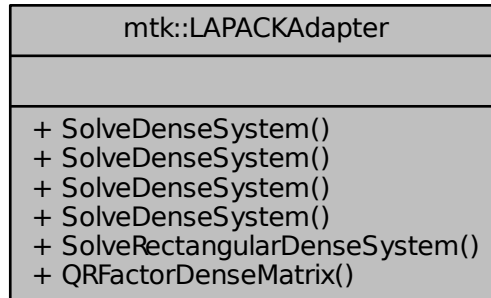
Static Public Member Functions

- 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_ld_)

  Solves overdetermined or underdetermined real linear systems.

- static mtk::DenseMatrix QRFactorDenseMatrix (DenseMatrix &matrix)

  Performs a QR factorization on a dense matrix.

16.16.1   Detailed Description

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

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

See also

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

Definition at line 93 of file mtk_lapack_adapter.h.

16.16.2 Member Function Documentation

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

Adapts the MTK to LAPACK's routine.

Parameters

in,out	*matrix*	Input matrix.

Returns

> Matrix **Q**.

Exceptions

std::bad_alloc	

Definition at line 593 of file mtk_lapack_adapter.cc.

Here is the call graph for this function:
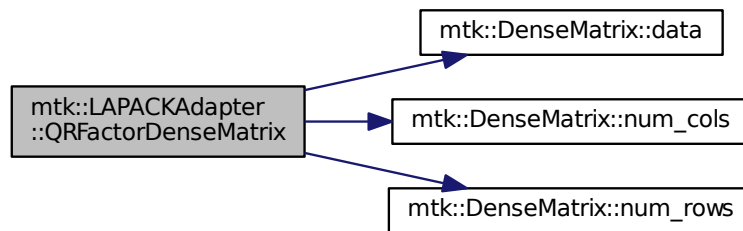
Here is the caller graph for this function:

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

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

in	*matrix*	Input matrix.
in	*rhs*	Input right-hand sides vector.

Exceptions

std::bad_alloc	

Definition at line 430 of file mtk_lapack_adapter.cc.

Here is the call graph for this function:
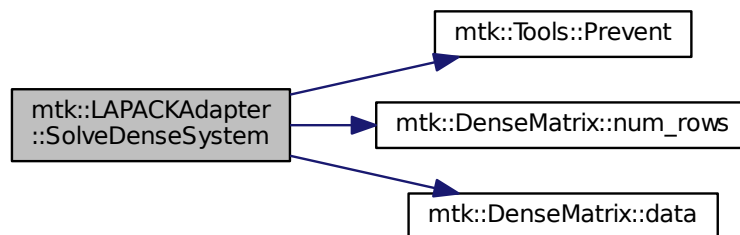
Here is the caller graph for this function:
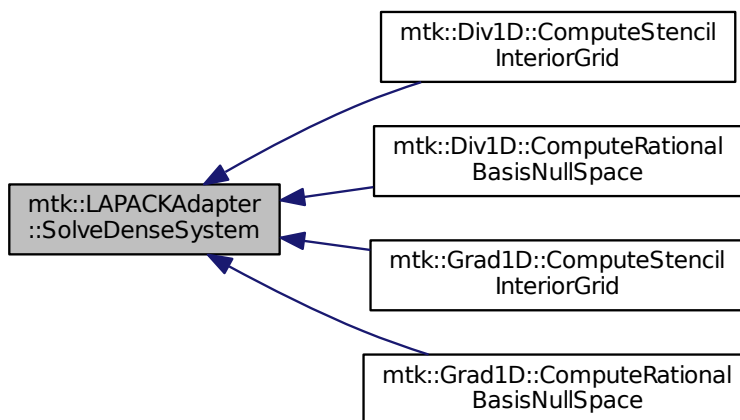
16.16.2.3   int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & *mm,* mtk::DenseMatrix & *rr*) `[static]`

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

in	*matrix*	Input matrix.
in	*rr*	Input right-hand sides matrix.

Exceptions

std::bad_alloc	

Definition at line 465 of file mtk_lapack_adapter.cc.

Here is the call graph for this function:
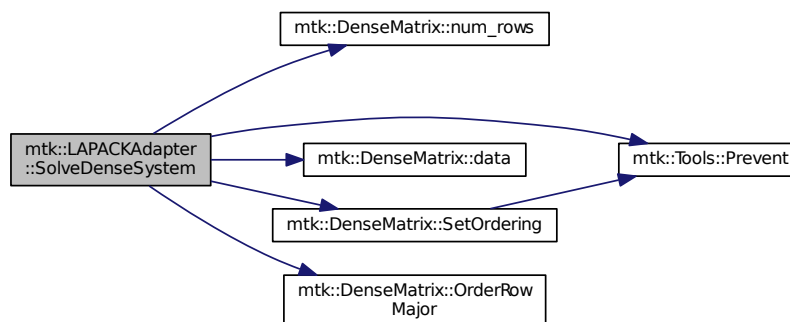
16.16.2.4   int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & *mm,* mtk::UniStgGrid1D & *rhs*)
`[static]`

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

in	*matrix*	Input matrix.
in	*rhs*	Input right-hand side from info on a grid.

Exceptions

std::bad_alloc	

Definition at line 517 of file mtk_lapack_adapter.cc.

Here is the call graph for this function:
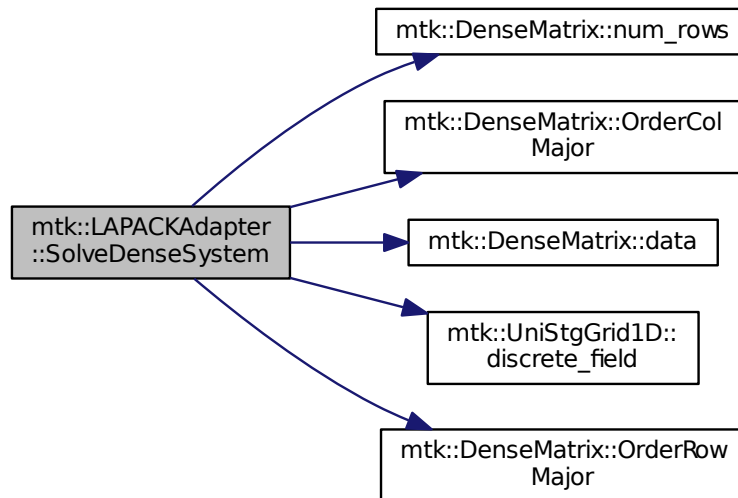
16.16.2.5   int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & *mm,* **mtk::UniStgGrid2D &** *rhs* **)**
`[static]`

Adapts the MTK to LAPACK's dgesv_ routine.

Parameters

in	*matrix*	Input matrix.
in	*rhs*	Input right-hand side from info on a grid.

Exceptions

std::bad_alloc	

Definition at line 555 of file mtk_lapack_adapter.cc.

Here is the call graph for this function:
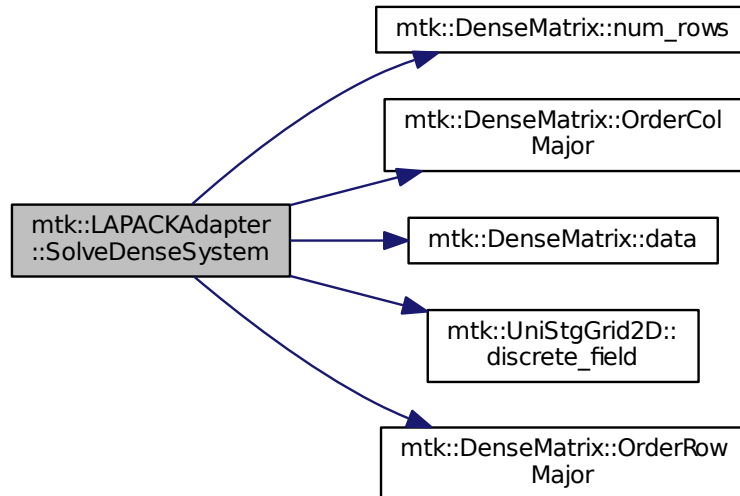
16.16.2.6   int mtk::LAPACKAdapter::SolveRectangularDenseSystem (const mtk::DenseMatrix & *aa,* mtk::Real ∗ *ob_,* int *ob_ld_*) `[static]`

Adapts the MTK to LAPACK's routine.

Parameters

`in,out`	*matrix*	Input matrix.

Returns

Success of the solution.

Exceptions

std::bad_alloc	

Definition at line 790 of file mtk_lapack_adapter.cc.
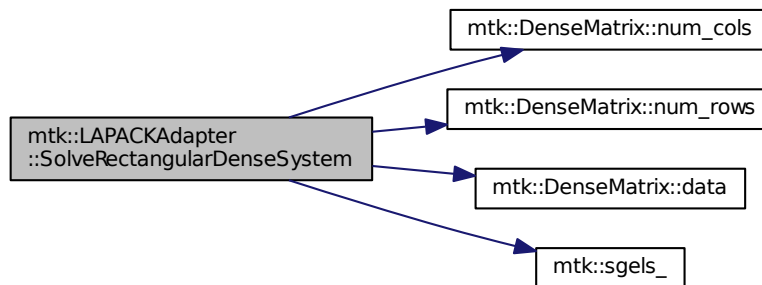
Here is the call graph for this function:

Here is the caller graph for this function:
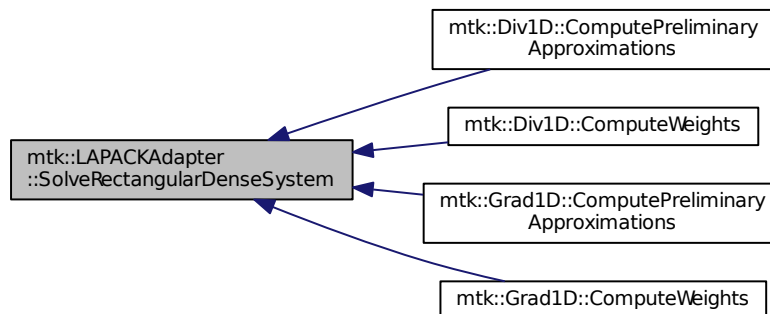
The documentation for this class was generated from the following files:

- include/mtk_lapack_adapter.h
- src/mtk_lapack_adapter.cc

16.17 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          │
├─────────────────────────────┤
│ - storage_                  │
│ - ordering_                 │
│ - num_rows_                 │
│ - num_cols_                 │
│ - num_values_               │
│ - ld_                       │
│ - num_zero_                 │
│ - num_non_zero_             │
│ - num_null_                 │
│ - num_non_null_             │
│ and 7 more...               │
├─────────────────────────────┤
│ + 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.

16.17.1 Detailed Description

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

Definition at line 75 of file mtk_matrix.h.

16.17.2 Constructor & Destructor Documentation

16.17.2.1 mtk::Matrix::Matrix ( )

Definition at line 67 of file mtk_matrix.cc.

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

Parameters

in		*in*	Given matrix.

Definition at line 86 of file mtk_matrix.cc.

16.17.2.3 mtk::Matrix::∼Matrix ( ) `[noexcept]`

Definition at line 105 of file mtk_matrix.cc.

16.17.3 Member Function Documentation

16.17.3.1 Real mtk::Matrix::abs_density ( ) const `[noexcept]`

See also

> http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

> Absolute density of the matrix.

16.17.3.2 mtk::Real mtk::Matrix::abs_sparsity ( ) const `[noexcept]`

See also

> http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

> Absolute sparsity of the matrix.

Definition at line 177 of file mtk_matrix.cc.

16.17.3.3 int mtk::Matrix::bandwidth ( ) const `[noexcept]`

Returns

> Bandwidth of the matrix.

Definition at line 167 of file mtk_matrix.cc.

16.17.3.4 void mtk::Matrix::IncreaseNumNull ( ) `[noexcept]`

Todo Review the definition of sparse matrices properties.

Definition at line 274 of file mtk_matrix.cc.

16.17.3.5   void mtk::Matrix::IncreaseNumZero (   ) `[noexcept]`

[Todo](#) Review the definition of sparse matrices properties.

Definition at line 264 of file mtk_matrix.cc.

16.17.3.6   int mtk::Matrix::kl (   ) const `[noexcept]`

Returns

> Number of lower diagonals.

Definition at line 157 of file mtk_matrix.cc.

16.17.3.7   int mtk::Matrix::ku (   ) const `[noexcept]`

Returns

> Number of upper diagonals.

Definition at line 162 of file mtk_matrix.cc.

16.17.3.8   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.

16.17.3.9   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:
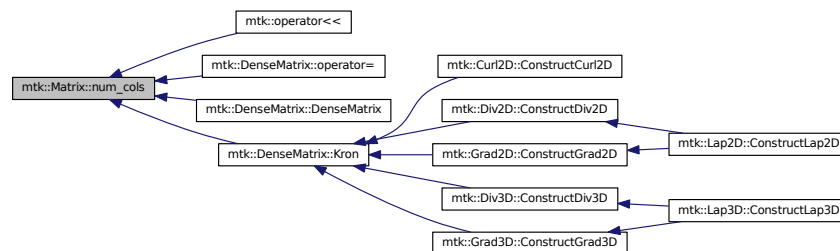
16.17.3.10 int mtk::Matrix::num_non_null ( ) const `[noexcept]`

See also

http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

Number of non-null values of the matrix.

Definition at line 152 of file mtk_matrix.cc.

16.17.3.11 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.

16.17.3.12 int mtk::Matrix::num_null ( ) const `[noexcept]`

See also

http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

Number of null values of the matrix.

Definition at line 147 of file mtk_matrix.cc.

Here is the caller graph for this function:
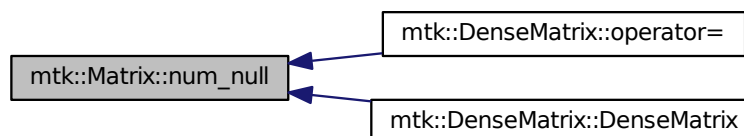
16.17.3.13 int mtk::Matrix::num_rows ( ) const `[noexcept]`

Returns

> Number of rows of the matrix.

Definition at line 117 of file mtk_matrix.cc.

Here is the caller graph for this function:

```
mtk::operator<<
mtk::Matrix::num_rows          mtk::DenseMatrix::operator=          mtk::Curl2D::ConstructCurl2D
                               mtk::DenseMatrix::DenseMatrix        mtk::Div2D::ConstructDiv2D          mtk::Lap2D::ConstructLap2D
                               mtk::DenseMatrix::Kron               mtk::Grad2D::ConstructGrad2D
                                                                    mtk::Div3D::ConstructDiv3D          mtk::Lap3D::ConstructLap3D
                                                                    mtk::Grad3D::ConstructGrad3D
```
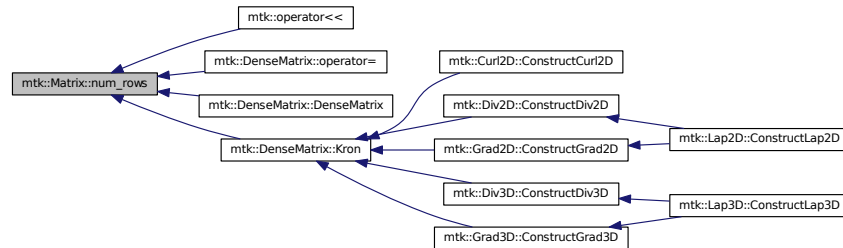
16.17.3.14   int mtk::Matrix::num_values (   ) const   `[noexcept]`

Returns

> Number of values of the matrix.

Definition at line 127 of file mtk_matrix.cc.

16.17.3.15   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:

```
                              mtk::DenseMatrix::operator=
mtk::Matrix::num_zero
                              mtk::DenseMatrix::DenseMatrix
```
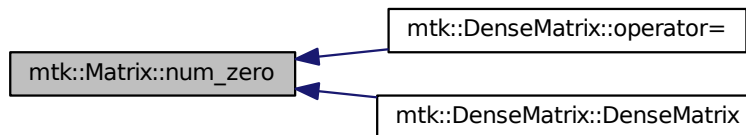
16.17.3.16   mtk::MatrixOrdering mtk::Matrix::ordering (   ) const   `[noexcept]`

Returns

Type of ordering of this matrix.

Definition at line 112 of file mtk_matrix.cc.

Here is the caller graph for this function:
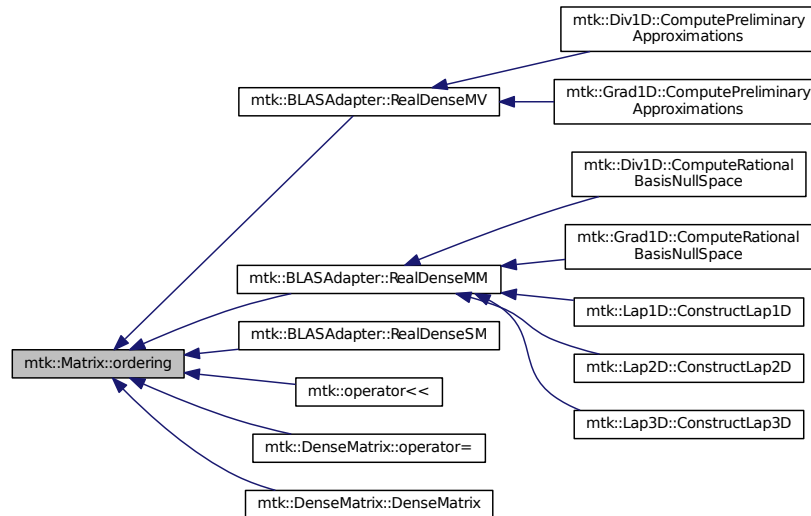
16.17.3.17   mtk::Real mtk::Matrix::rel_density ( ) const `[noexcept]`

See also

http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

Relative density of the matrix.

Definition at line 172 of file mtk_matrix.cc.

16.17.3.18   mtk::Real mtk::Matrix::rel_sparsity ( ) const `[noexcept]`

See also

http://www.csrc.sdsu.edu/research_reports/CSRCR2013-01.pdf

Returns

Relative sparsity of the matrix.

Definition at line 182 of file mtk_matrix.cc.

16.17.3.19   void mtk::Matrix::set_num_cols (const int & *num_cols*) `[noexcept]`

Parameters

in	*num_cols*	Number of columns.

Definition at line 224 of file mtk_matrix.cc.

Here is the call graph for this function:

```
┌─────────────────────────┐      ┌──────────────────────┐
│ mtk::Matrix::set_num_cols│─────▶│ mtk::Tools::Prevent  │
└─────────────────────────┘      └──────────────────────┘
```

Here is the caller graph for this function:

```
┌─────────────────────────┐      ┌────────────────────────────┐
│ mtk::Matrix::set_num_cols│◀─────│ mtk::DenseMatrix::operator= │
└─────────────────────────┘      └────────────────────────────┘
```

16.17.3.20   void mtk::Matrix::set_num_null (const int & *in*)   `[noexcept]`

Parameters

in	*in*	Number of zero values.

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

Definition at line 250 of file mtk_matrix.cc.

Here is the call graph for this function:

```
┌─────────────────────────┐      ┌──────────────────────┐
│ mtk::Matrix::set_num_null│─────▶│ mtk::Tools::Prevent  │
└─────────────────────────┘      └──────────────────────┘
```
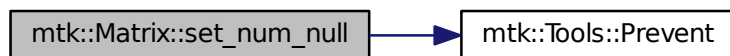
Here is the caller graph for this function:

16.17.3.21 **void mtk::Matrix::set_num_rows (const int & *num_rows*)** `[noexcept]`

Parameters

in	*num_rows*	Number of rows.

Definition at line 212 of file mtk_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

16.17.3.22 **void mtk::Matrix::set_num_zero (const int & *in*)** `[noexcept]`

Parameters

in	*in*	Number of zero values.

Bug -nan assigned on construction time due to num_values_ being 0.
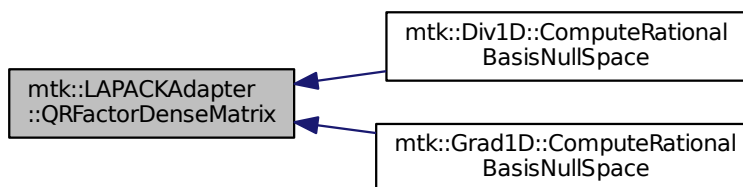
Definition at line 236 of file mtk_matrix.cc.

Here is the call graph for this function:
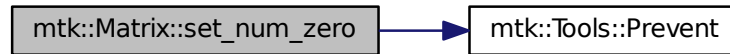
Here is the caller graph for this function:

16.17.3.23   void mtk::Matrix::set_ordering (const MatrixOrdering & *oo*)   `[noexcept]`

See also

    MatrixOrdering

Parameters

in	*oo*	Ordering of the matrix.

Definition at line 199 of file mtk_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:
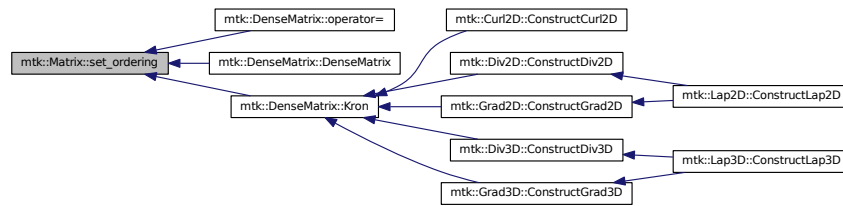
16.17.3.24   void mtk::Matrix::set_storage (const MatrixStorage & *tt*)   `[noexcept]`

See also

[MatrixStorage](#)

Parameters

`in`	*tt*	Type of the matrix storage.

Definition at line 187 of file mtk_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:
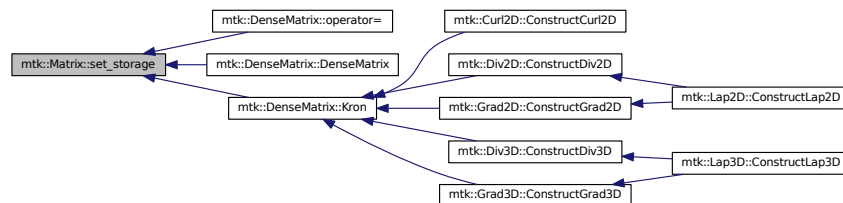
16.17.3.25   mtk::MatrixStorage mtk::Matrix::storage ( ) const   `[noexcept]`

Returns

Type of storage of this matrix.

Definition at line 107 of file mtk_matrix.cc.

Here is the caller graph for this function:
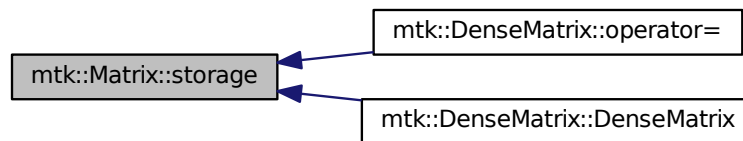
16.17.4 Member Data Documentation

16.17.4.1 Real mtk::Matrix::abs_density_ `[private]`

Definition at line 296 of file mtk_matrix.h.

16.17.4.2 Real mtk::Matrix::abs_sparsity_ `[private]`

Definition at line 298 of file mtk_matrix.h.

16.17.4.3 int mtk::Matrix::bandwidth_ `[private]`

Definition at line 294 of file mtk_matrix.h.

16.17.4.4 int mtk::Matrix::kl_ `[private]`

Definition at line 292 of file mtk_matrix.h.

16.17.4.5 int mtk::Matrix::ku_ `[private]`

Definition at line 293 of file mtk_matrix.h.

16.17.4.6 int mtk::Matrix::ld_ `[private]`

Definition at line 285 of file mtk_matrix.h.

16.17.4.7 int mtk::Matrix::num_cols_ `[private]`

Definition at line 283 of file mtk_matrix.h.

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

Definition at line 290 of file mtk_matrix.h.

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

Definition at line 288 of file mtk_matrix.h.

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

Definition at line 289 of file mtk_matrix.h.

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

Definition at line 282 of file mtk_matrix.h.

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

Definition at line 284 of file mtk_matrix.h.

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

Definition at line 287 of file mtk_matrix.h.

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

Definition at line 280 of file mtk_matrix.h.

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

Definition at line 297 of file mtk_matrix.h.

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

Definition at line 299 of file mtk_matrix.h.

16.17.4.17 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

16.18   mtk::Quad1D Class Reference

Implements a 1D mimetic quadrature.

`#include <mtk_quad_1d.h>`

Collaboration diagram for mtk::Quad1D:
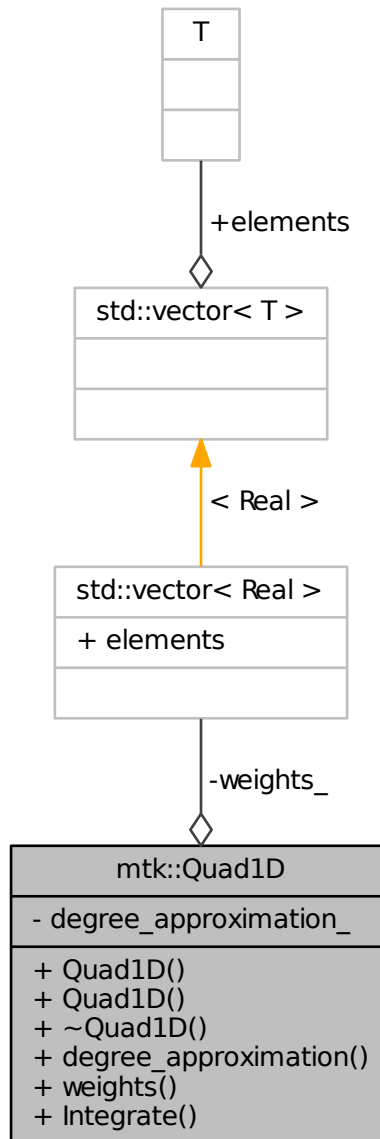
Public Member Functions

- 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.

16.18.1   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.

16.18.2   Constructor & Destructor Documentation

16.18.2.1   mtk::Quad1D::Quad1D (   )

16.18.2.2   mtk::Quad1D::Quad1D (const Quad1D & *quad*)

Parameters

in	*div*	Given quadrature.

16.18.2.3   mtk::Quad1D::∼Quad1D (   )

16.18.3   Member Function Documentation

16.18.3.1   int mtk::Quad1D::degree_approximation (   ) const

Returns

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

16.18.3.2   Real mtk::Quad1D::Integrate (Real(∗)(Real xx) *Integrand,* UniStgGrid1D *grid*) const

Parameters

in	*Integrand*	Real-valued function to integrate.
in	*grid*	Given integration domain.

Returns

Result of the integration.

16.18.3.3   Real∗ mtk::Quad1D::weights (   ) const

Returns

Collection of weights.

16.18.4   Friends And Related Function Documentation

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

16.18.5   Member Data Documentation

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

Definition at line 124 of file mtk_quad_1d.h.

16.18.5.2   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

16.19   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:
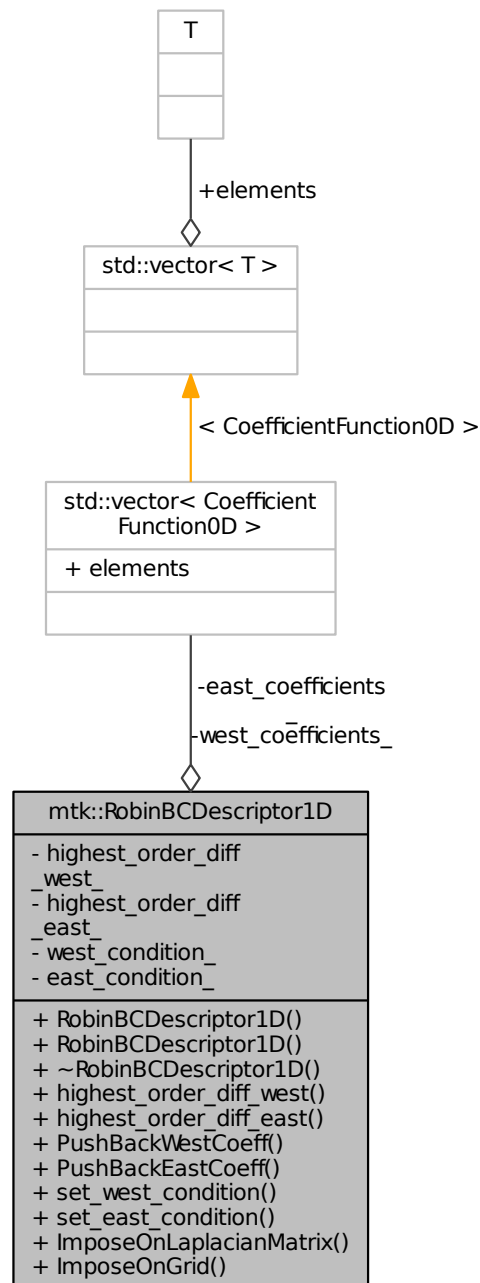
Public Member Functions

- 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.

16.19.1   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.

16.19.2 Constructor & Destructor Documentation

16.19.2.1 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D ( )

Definition at line 93 of file mtk_robin_bc_descriptor_1d.cc.

16.19.2.2 mtk::RobinBCDescriptor1D::RobinBCDescriptor1D (const **RobinBCDescriptor1D** & *desc*)

Parameters

in	*desc*	Given 1D descriptor.

Definition at line 99 of file mtk_robin_bc_descriptor_1d.cc.

16.19.2.3 mtk::RobinBCDescriptor1D::∼RobinBCDescriptor1D ( ) `[noexcept]`

Definition at line 106 of file mtk_robin_bc_descriptor_1d.cc.

16.19.3 Member Function Documentation

16.19.3.1 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.

16.19.3.2 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.

16.19.3.3   void mtk::RobinBCDescriptor1D::ImposeOnGrid (UniStgGrid1D & *grid,* const Real & *time =* mtk::kZero) const

Parameters

in,out	*grid*	Grid upon which impose the desired boundary condition.
in	*time*	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:
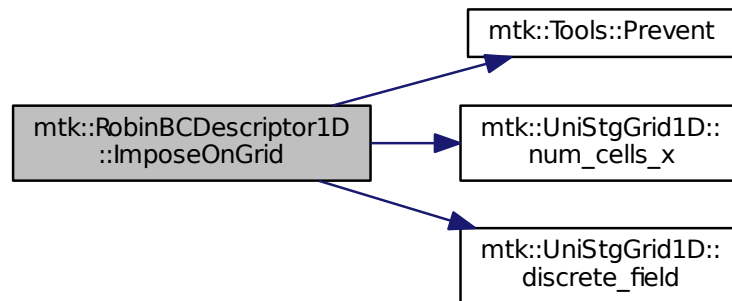
16.19.3.4   bool mtk::RobinBCDescriptor1D::ImposeOnLaplacianMatrix (const Lap1D & *lap,* mtk::DenseMatrix & *matrix,* const Real & *time =* mtk::kZero) const

Parameters

in	*lap*	Operator in the Matrix.
in,out	*matrix*	Input Laplacian operator.
in	*time*	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:
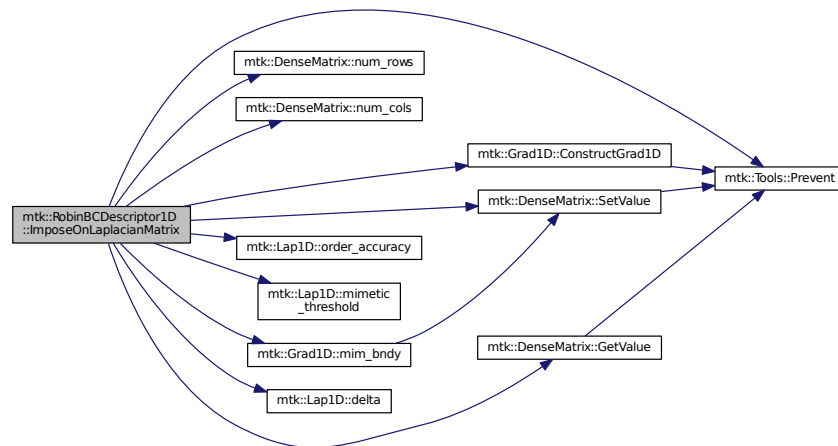
16.19.3.5   void mtk::RobinBCDescriptor1D::PushBackEastCoeff (mtk::CoefficientFunction0D *ce*)

Parameters

in		*ce*	Function $c_e(x, y) : \Omega \mapsto \mathbb{R}$.

Definition at line 132 of file mtk_robin_bc_descriptor_1d.cc.

Here is the call graph for this function:
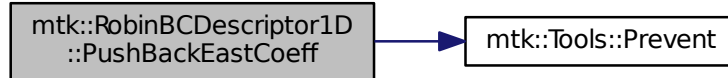
16.19.3.6 void mtk::RobinBCDescriptor1D::PushBackWestCoeff (mtk::CoefficientFunction0D *cw*)

Parameters

in		*cw*	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:
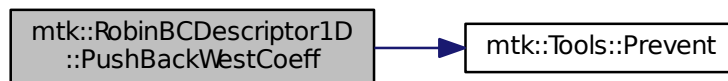
16.19.3.7 void mtk::RobinBCDescriptor1D::set_east_condition (Real(∗)(const Real &tt) *east_condition*) `[noexcept]`

Parameters

in		*east_condition*	$\beta_e(y, t) : \Omega \mapsto \mathbb{R}$.

Definition at line 156 of file mtk_robin_bc_descriptor_1d.cc.

Here is the call graph for this function:
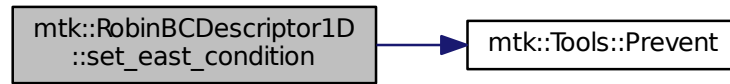
16.19.3.8   void mtk::RobinBCDescriptor1D::set_west_condition (Real(∗)(const Real &tt) *west_condition*) `[noexcept]`

Parameters

in	*west_condition*	$\beta_w(y,t) : \Omega \mapsto \mathbb{R}$.

Definition at line 146 of file mtk_robin_bc_descriptor_1d.cc.

Here is the call graph for this function:
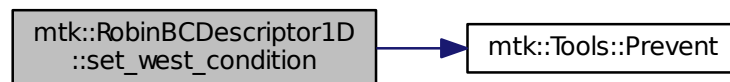
16.19.4   Member Data Documentation

16.19.4.1   std::vector<CoefficientFunction0D> mtk::RobinBCDescriptor1D::east_coefficients_ `[private]`

Definition at line 237 of file mtk_robin_bc_descriptor_1d.h.

16.19.4.2   Real(∗ mtk::RobinBCDescriptor1D::east_condition_)(const Real &tt) `[private]`

Definition at line 240 of file mtk_robin_bc_descriptor_1d.h.

16.19.4.3   int mtk::RobinBCDescriptor1D::highest_order_diff_east_ `[private]`

Definition at line 234 of file mtk_robin_bc_descriptor_1d.h.

16.19.4.4   int mtk::RobinBCDescriptor1D::highest_order_diff_west_ `[private]`

Definition at line 233 of file mtk_robin_bc_descriptor_1d.h.

16.19.4.5 std::vector<**CoefficientFunction0D**> **mtk::RobinBCDescriptor1D::west_coefficients_** [private]

Definition at line 236 of file mtk_robin_bc_descriptor_1d.h.

16.19.4.6 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

16.20 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:
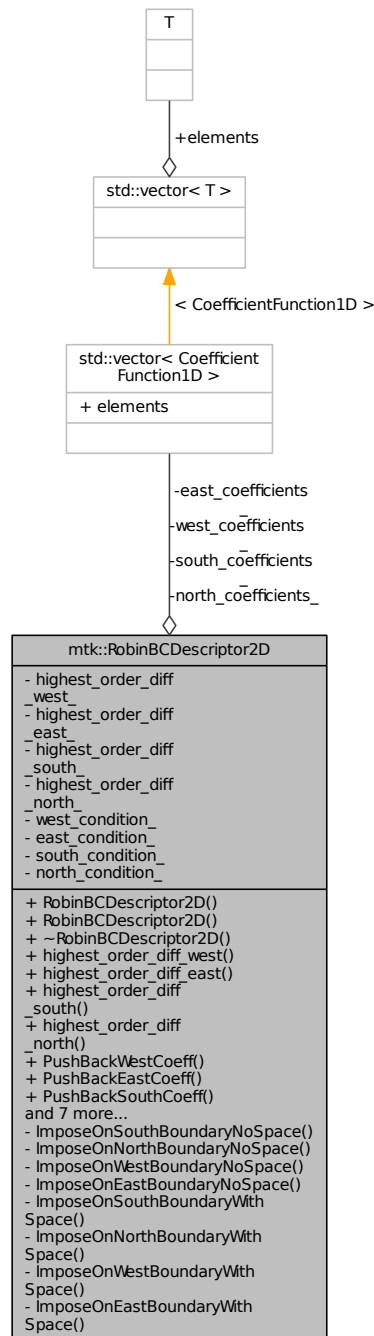
Public Member Functions

- 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.*

16.20.1 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.

16.20.2 Constructor & Destructor Documentation

16.20.2.1 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D ( )

Definition at line 84 of file mtk_robin_bc_descriptor_2d.cc.

16.20.2.2 mtk::RobinBCDescriptor2D::RobinBCDescriptor2D (const RobinBCDescriptor2D & *desc*)

Parameters

in	*desc*	Given 2D descriptor.

Definition at line 94 of file mtk_robin_bc_descriptor_2d.cc.

16.20.2.3 mtk::RobinBCDescriptor2D::∼RobinBCDescriptor2D ( ) `[noexcept]`

Definition at line 105 of file mtk_robin_bc_descriptor_2d.cc.

16.20.3 Member Function Documentation

16.20.3.1 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.

16.20.3.2 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.

16.20.3.3 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.

16.20.3.4 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.

16.20.3.5 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time* **= kZero) const** `[private]`

Parameters

`in`	*lap*	Laplacian operator on the matrix.
`in`	*grid*	Grid upon which impose the desired boundary condition.
`in,out`	*matrix*	Input matrix with the Laplacian operator.
`in`	*time*	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:

16.20.3.6   bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryWithSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero   ) const**   [private]

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:
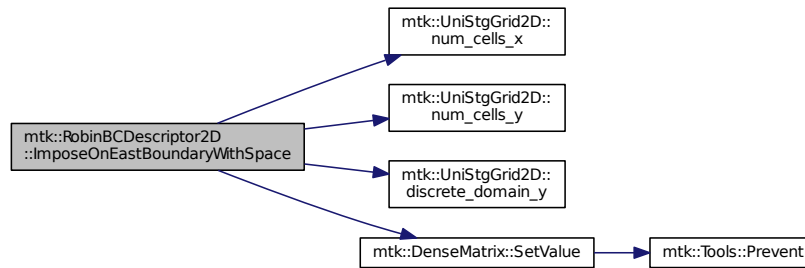
16.20.3.7 void mtk::RobinBCDescriptor2D::ImposeOnGrid (mtk::UniStgGrid2D & *grid,* const Real & *time =* kZero) const

Parameters

in,out	*grid*	Grid upon which impose the desired boundary condition.
in	*time*	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:

16.20.3.8 bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const**

Parameters

in	lap	Laplacian operator on the matrix.
in	grid	Grid upon which impose the desired boundary condition.
in,out	matrix	Input matrix with the Laplacian operator.
in	time	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:

16.20.3.9 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:

16.20.3.10 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace (const **Lap2D** & *lap,* const **UniStgGrid2D** & *grid,* **mtk::DenseMatrix** & *matrix,* const **Real** & *time* = **kZero**) const [private]

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:

16.20.3.11 bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in	lap	Laplacian operator on the matrix.
in	grid	Grid upon which impose the desired boundary condition.
in,out	matrix	Input matrix with the Laplacian operator.
in	time	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:

```
mtk::RobinBCDescriptor2D          mtk::UniStgGrid2D::
::ImposeOnSouthBoundaryNoSpace        west_bndy

                                  mtk::DenseMatrix::num_rows

                                  mtk::DenseMatrix::num_cols

                                  mtk::DenseMatrix::SetValue ──── mtk::Tools::Prevent

                                  mtk::UniStgGrid2D::
                                      delta_x

                                  mtk::UniStgGrid2D::
                                      num_cells_x

                                  mtk::UniStgGrid2D::
                                      east_bndy
```

16.20.3.12   bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace (const Lap2D & *lap,* const UniStgGrid2D & *grid,* mtk::DenseMatrix & *matrix,* const Real & *time* = kZero) const   [private]

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:

```
                                  mtk::UniStgGrid2D::
                                      num_cells_x

mtk::RobinBCDescriptor2D          mtk::UniStgGrid2D::
::ImposeOnSouthBoundaryWithSpace      discrete_domain_x

                                  mtk::DenseMatrix::SetValue ──── mtk::Tools::Prevent
```
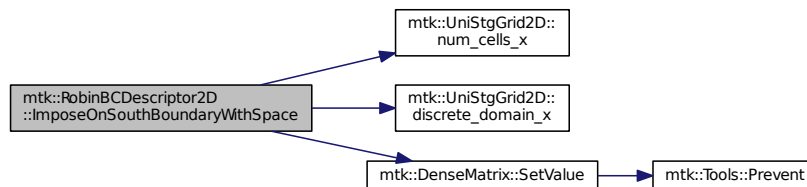
16.20.3.13 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:
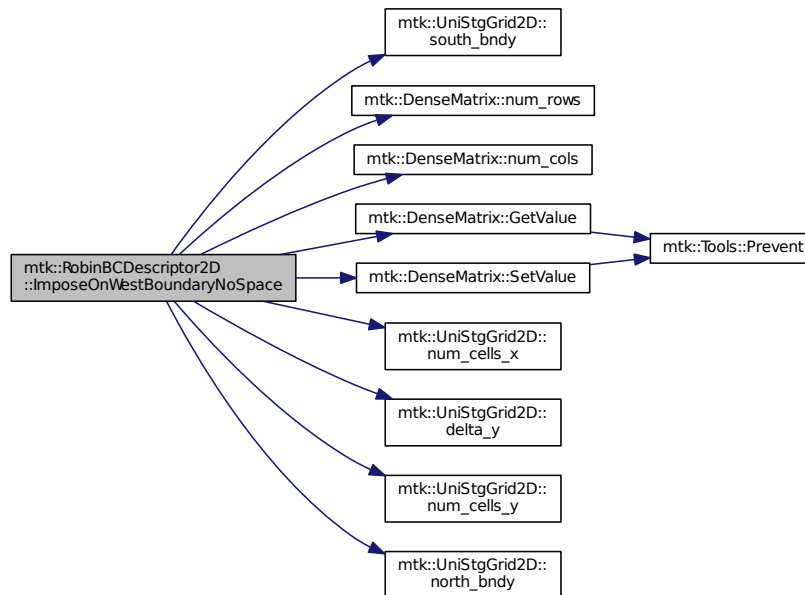
16.20.3.14 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryWithSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **mtk::DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	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:
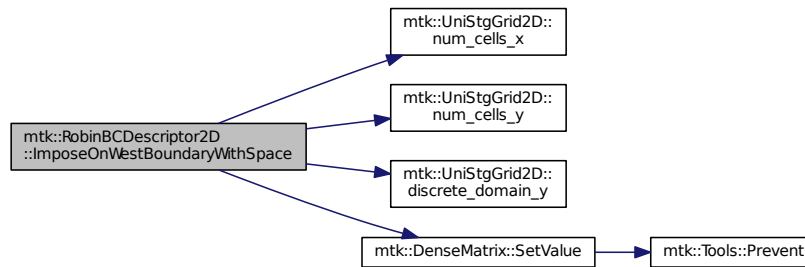
16.20.3.15 void mtk::RobinBCDescriptor2D::PushBackEastCoeff (mtk::CoefficientFunction1D *ce*)

Parameters

in	*cw*	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:
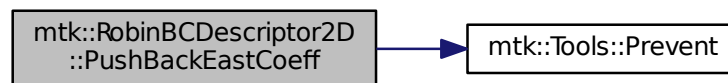
16.20.3.16 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff (mtk::CoefficientFunction1D *cn*)

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:
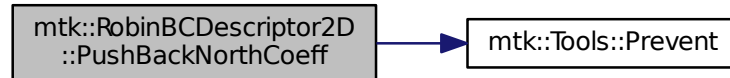
16.20.3.17   void mtk::RobinBCDescriptor2D::PushBackSouthCoeff (mtk::CoefficientFunction1D *cs*)

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:
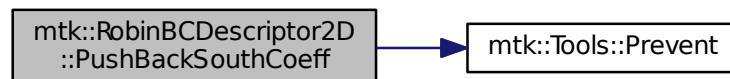
16.20.3.18   void mtk::RobinBCDescriptor2D::PushBackWestCoeff (mtk::CoefficientFunction1D *cw*)

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:
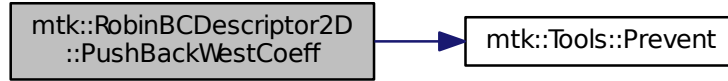
16.20.3.19 void mtk::RobinBCDescriptor2D::set_east_condition (Real(∗)(const Real &yy, const Real &tt) *east_condition*)
[noexcept]

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:
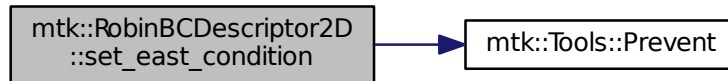
16.20.3.20 void mtk::RobinBCDescriptor2D::set_north_condition (Real(∗)(const Real &xx, const Real &tt) *north_condition*)
[noexcept]

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:
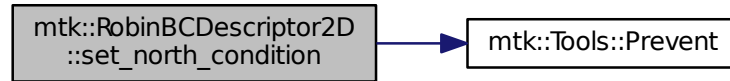
16.20.3.21   void mtk::RobinBCDescriptor2D::set_south_condition (Real(∗)(const **Real &xx, const Real &tt)** *south_condition* **)**
    `[noexcept]`

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:
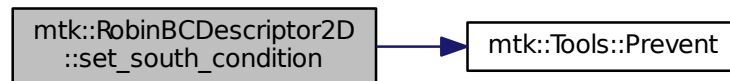
16.20.3.22   void mtk::RobinBCDescriptor2D::set_west_condition (Real(∗)(const **Real &yy, const Real &tt)** *west_condition* **)**
    `[noexcept]`

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:
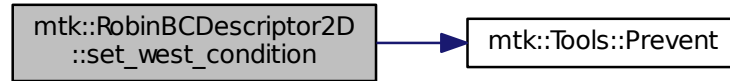
16.20.4 Member Data Documentation

16.20.4.1 std::vector<**CoefficientFunction1D**> mtk::RobinBCDescriptor2D::east_coefficients_ `[private]`

Definition at line 367 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.2 Real(∗ mtk::RobinBCDescriptor2D::east_condition_)(const Real &xx, const Real &tt) `[private]`

Definition at line 372 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.3 int mtk::RobinBCDescriptor2D::highest_order_diff_east_ `[private]`

Definition at line 362 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.4 int mtk::RobinBCDescriptor2D::highest_order_diff_north_ `[private]`

Definition at line 364 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.5 int mtk::RobinBCDescriptor2D::highest_order_diff_south_ `[private]`

Definition at line 363 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.6 int mtk::RobinBCDescriptor2D::highest_order_diff_west_ `[private]`

Definition at line 361 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.7 std::vector<**CoefficientFunction1D**> mtk::RobinBCDescriptor2D::north_coefficients_ `[private]`

Definition at line 369 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.8 Real(∗ mtk::RobinBCDescriptor2D::north_condition_)(const Real &yy, const Real &tt) `[private]`

Definition at line 374 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.9 std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::south_coefficients_ `[private]`

Definition at line 368 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.10 Real(∗ mtk::RobinBCDescriptor2D::south_condition_)(const Real &yy, const Real &tt) `[private]`

Definition at line 373 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.11 std::vector<CoefficientFunction1D> mtk::RobinBCDescriptor2D::west_coefficients_ `[private]`

Definition at line 366 of file mtk_robin_bc_descriptor_2d.h.

16.20.4.12 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

16.21 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:

```
                          ┌──────────┐
                          │    T     │
                          ├──────────┤
                          │          │
                          └──────────┘
                               │
                               │ +elements
                               ◇
                    ┌────────────────────┐
                    │  std::vector< T >  │
                    ├────────────────────┤
                    │                    │
                    ├────────────────────┤
                    │                    │
                    └────────────────────┘
                               ▲
                               │ < CoefficientFunction2D >
                    ┌────────────────────┐
                    │ std::vector< Coefficient │
                    │     Function2D >   │
                    ├────────────────────┤
                    │  + elements        │
                    ├────────────────────┤
                    │                    │
                    └────────────────────┘
                               │
                               │  -east_coefficients
                               │  -west_coefficients
                               │  -south_coefficients
                               │  -top_coefficients
                               │  -bottom_coefficients
                               │  -north_coefficients_
                               ◇
```

```
┌──────────────────────────────────────┐
│       mtk::RobinBCDescriptor3D        │
├──────────────────────────────────────┤
│ - highest_order_diff                  │
│ _west_                                │
│ - highest_order_diff                  │
│ _east_                                │
│ - highest_order_diff                  │
│ _south_                               │
│ - highest_order_diff                  │
│ _north_                               │
│ - highest_order_diff                  │
│ _bottom_                              │
│ - highest_order_diff_top_             │
│ - west_condition_                     │
│ - east_condition_                     │
│ - south_condition_                    │
│ - north_condition_                    │
│ - bottom_condition_                   │
│ - top_condition_                      │
├──────────────────────────────────────┤
│ + RobinBCDescriptor3D()               │
│ + RobinBCDescriptor3D()               │
│ + ~RobinBCDescriptor3D()              │
│ + highest_order_diff_west()           │
│ + PushBackWestCoeff()                 │
│ + set_west_condition()                │
│ + ImposeOnLaplacianMatrix()           │
│ + ImposeOnGrid()                      │
│ - ImposeOnSouthBoundaryNoSpace()      │
│ - ImposeOnNorthBoundaryNoSpace()      │
│ - ImposeOnWestBoundaryNoSpace()       │
│ - ImposeOnEastBoundaryNoSpace()       │
│ - ImposeOnSouthBoundaryWith           │
│ Space()                               │
│ - ImposeOnNorthBoundaryWith           │
│ Space()                               │
│ - ImposeOnWestBoundaryWith            │
│ Space()                               │
│ - ImposeOnEastBoundaryWith            │
│ Space()                               │
└──────────────────────────────────────┘
```

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
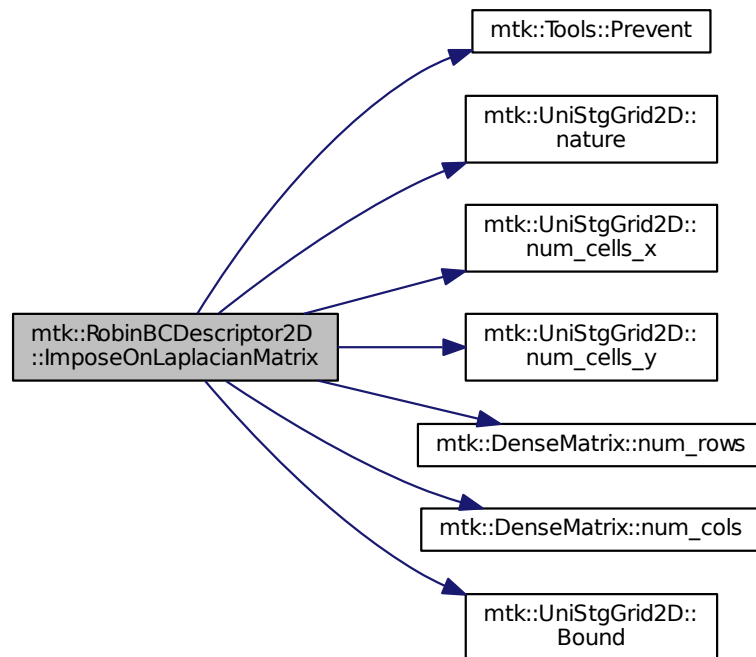
    Imposes the condition on the operator represented as matrix.

- 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
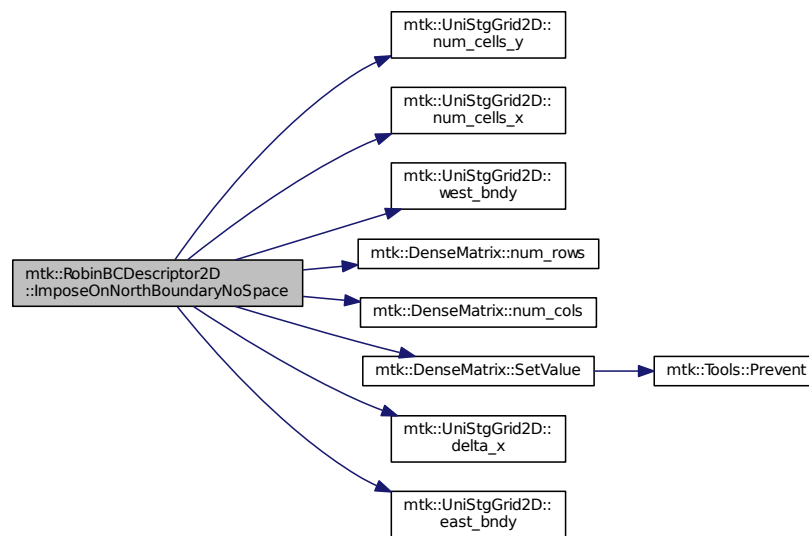
    Imposes the condition on the north boundary.

- 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.

16.21.1 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.

16.21.2 Constructor & Destructor Documentation

16.21.2.1 mtk::RobinBCDescriptor3D::RobinBCDescriptor3D ( )

16.21.2.2 mtk::RobinBCDescriptor3D::RobinBCDescriptor3D (const **RobinBCDescriptor3D** & *desc*)

Parameters

in	*desc*	Given 2D descriptor.

16.21.2.3 mtk::RobinBCDescriptor3D::∼RobinBCDescriptor3D ( ) `[noexcept]`

16.21.3 Member Function Documentation

16.21.3.1 int mtk::RobinBCDescriptor3D::highest_order_diff_west ( ) const `[noexcept]`

Returns

Integer highest order of differentiation in the ∗ face.

16.21.3.2 bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryNoSpace (const **Lap2D** & *lap,* const **UniStgGrid2D** & *grid,* **DenseMatrix** & *matrix,* const **Real** & *time* = **kZero**) const `[private]`

Parameters

in		*lap*	Laplacian operator on the matrix.
in		*grid*	Grid upon which impose the desired boundary condition.
in,out		*matrix*	Input matrix with the Laplacian operator.
in		*time*	Current time snapshot. Default is kZero.

16.21.3.3 bool mtk::RobinBCDescriptor3D::ImposeOnEastBoundaryWithSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in		*lap*	Laplacian operator on the matrix.
in		*grid*	Grid upon which impose the desired boundary condition.
in,out		*matrix*	Input matrix with the Laplacian operator.
in		*time*	Current time snapshot. Default is kZero.

16.21.3.4   void mtk::RobinBCDescriptor3D::ImposeOnGrid (UniStgGrid3D & *grid,* **const Real &** *time =* **kZero) const**

Parameters

in,out		*grid*	Grid upon which impose the desired boundary condition.
in		*time*	Current time snapshot. Default is kZero.

16.21.3.5   bool mtk::RobinBCDescriptor3D::ImposeOnLaplacianMatrix (const Lap3D & *lap,* **const UniStgGrid3D &** *grid,* **DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const**

Parameters

in		*lap*	Laplacian operator on the matrix.
in		*grid*	Grid upon which impose the desired boundary condition.
in,out		*matrix*	Input matrix with the Laplacian operator.
in		*time*	Current time snapshot. Default is kZero.

16.21.3.6   bool mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryNoSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in		*lap*	Laplacian operator on the matrix.
in		*grid*	Grid upon which impose the desired boundary condition.
in,out		*matrix*	Input matrix with the Laplacian operator.
in		*time*	Current time snapshot. Default is kZero.

16.21.3.7   bool mtk::RobinBCDescriptor3D::ImposeOnNorthBoundaryWithSpace (const Lap2D & *lap,* **const UniStgGrid2D &** *grid,* **DenseMatrix &** *matrix,* **const Real &** *time =* **kZero) const** `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	Current time snapshot. Default is kZero.

16.21.3.8 bool mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryNoSpace (const Lap2D & *lap,* const UniStgGrid2D & *grid,* DenseMatrix & *matrix,* const Real & *time =* kZero) const `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	Current time snapshot. Default is kZero.

16.21.3.9 bool mtk::RobinBCDescriptor3D::ImposeOnSouthBoundaryWithSpace (const Lap2D & *lap,* const UniStgGrid2D & *grid,* DenseMatrix & *matrix,* const Real & *time =* kZero) const `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	Current time snapshot. Default is kZero.

16.21.3.10 bool mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryNoSpace (const Lap2D & *lap,* const UniStgGrid2D & *grid,* DenseMatrix & *matrix,* const Real & *time =* kZero) const `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.
in	*time*	Current time snapshot. Default is kZero.

16.21.3.11 bool mtk::RobinBCDescriptor3D::ImposeOnWestBoundaryWithSpace (const Lap2D & *lap,* const UniStgGrid2D & *grid,* DenseMatrix & *matrix,* const Real & *time =* kZero) const `[private]`

Parameters

in	*lap*	Laplacian operator on the matrix.
in	*grid*	Grid upon which impose the desired boundary condition.
in,out	*matrix*	Input matrix with the Laplacian operator.

in	*time*	Current time snapshot. Default is kZero.

16.21.3.12   void mtk::RobinBCDescriptor3D::PushBackWestCoeff (CoefficientFunction2D *cw*)

Parameters

in	*cw*	Coeff. $c_w(x,y,t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.

16.21.3.13   void mtk::RobinBCDescriptor3D::set_west_condition (Real(∗)(const Real &xx, const Real &yy, const Real &tt) *west_condition*) `[noexcept]`

Parameters

in	*west_condition*	$\beta_w(x,y,t) : \partial\Omega \times [t_0, t_n] \mapsto \mathbb{R}$.

16.21.4   Member Data Documentation

16.21.4.1   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::bottom_coefficients_ `[private]`

Definition at line 309 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.2   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.

16.21.4.3   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::east_coefficients_ `[private]`

Definition at line 306 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.4   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.

16.21.4.5   int mtk::RobinBCDescriptor3D::highest_order_diff_bottom_ `[private]`

Definition at line 302 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.6   int mtk::RobinBCDescriptor3D::highest_order_diff_east_ `[private]`

Definition at line 299 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.7   int mtk::RobinBCDescriptor3D::highest_order_diff_north_ `[private]`

Definition at line 301 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.8   int mtk::RobinBCDescriptor3D::highest_order_diff_south_   `[private]`

Definition at line 300 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.9   int mtk::RobinBCDescriptor3D::highest_order_diff_top_   `[private]`

Definition at line 303 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.10   int mtk::RobinBCDescriptor3D::highest_order_diff_west_   `[private]`

Definition at line 298 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.11   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::north_coefficients_   `[private]`

Definition at line 308 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.12   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.

16.21.4.13   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::south_coefficients_   `[private]`

Definition at line 307 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.14   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.

16.21.4.15   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::top_coefficients_   `[private]`

Definition at line 310 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.16   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.

16.21.4.17   std::vector<CoefficientFunction2D> mtk::RobinBCDescriptor3D::west_coefficients_   `[private]`

Definition at line 305 of file mtk_robin_bc_descriptor_3d.h.

16.21.4.18 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

16.22 mtk::Tools Class Reference

Tool manager class.

`#include <mtk_tools.h>`

Collaboration diagram for mtk::Tools:
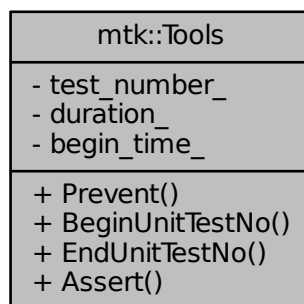
Static Public Member Functions

- 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.

16.22.1 Detailed Description

Basic tools to ensure execution correctness.

Definition at line 78 of file mtk_tools.h.

16.22.2 Member Function Documentation

16.22.2.1 void mtk::Tools::Assert (const bool & *condition*) `[static],[noexcept]`

Parameters

in	*condition*	Condition to be asserted.

Definition at line 109 of file mtk_tools.cc.

16.22.2.2 void mtk::Tools::BeginUnitTestNo (const int & *nn*) `[static],[noexcept]`

Parameters

in	*nn*	Number of the test.

Definition at line 88 of file mtk_tools.cc.

Here is the call graph for this function:
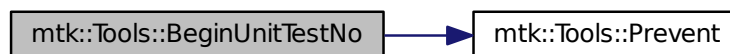
16.22.2.3 void mtk::Tools::EndUnitTestNo (const int & *nn*) `[static],[noexcept]`

Parameters

in	*nn*	Number of the test.

Definition at line 100 of file mtk_tools.cc.

Here is the call graph for this function:
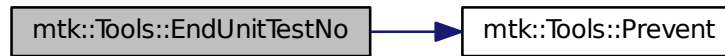
16.22.2.4 **void mtk::Tools::Prevent (const bool** *complement,* **const char** ∗**const** *fname,* **int** *lineno,* **const char** ∗**const** *fxname* **)** `[static],[noexcept]`

See also

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

Parameters

in	*complement*	Complement of desired pre-condition.
in	*fname*	Name of the file being checked.
in	*lineno*	Number of the line where the check is executed.
in	*fxname*	Name of the module containing the check.

Todo Check if this is the best way of stalling execution.

Definition at line 61 of file mtk_tools.cc.

16.22.3 **Member Data Documentation**

16.22.3.1 **clock_t mtk::Tools::begin_time_** `[static],[private]`

Definition at line 121 of file mtk_tools.h.

16.22.3.2 **mtk::Real mtk::Tools::duration_** `[static],[private]`

Definition at line 119 of file mtk_tools.h.

16.22.3.3 **int mtk::Tools::test_number_** `[static],[private]`

Todo Check usage of static methods and private members.

Definition at line 117 of file mtk_tools.h.

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

- include/mtk_tools.h
- src/mtk_tools.cc

16.23   mtk::UniStgGrid1D Class Reference

Uniform 1D Staggered Grid.

`#include <mtk_uni_stg_grid_1d.h>`

Collaboration diagram for mtk::UniStgGrid1D:
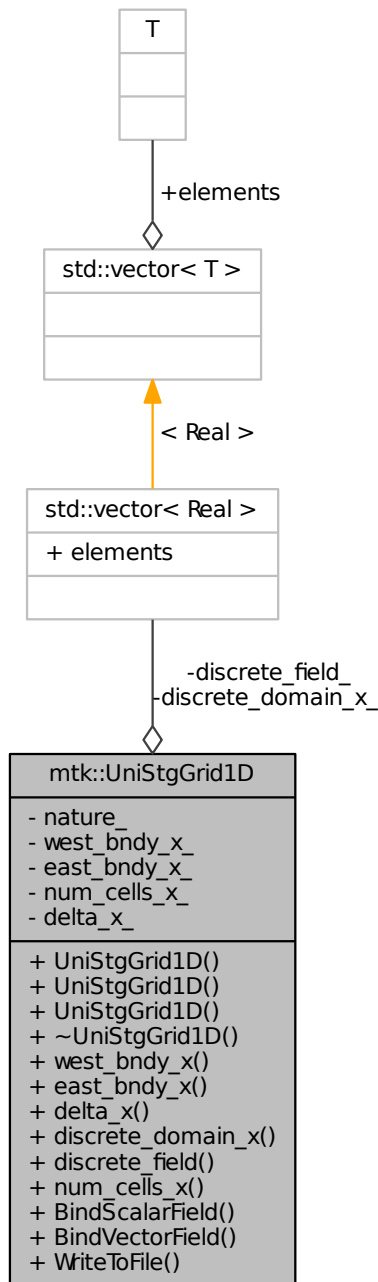
Public Member Functions

- 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 &nature=mtk::SCALAR)

    Construct a grid based on spatial discretization parameters.
- ∼UniStgGrid1D ()

    Destructor.
- Real west_bndy_x () const

    Provides access to west boundary spatial coordinate.
- Real east_bndy_x () const

    Provides access to east boundary spatial coordinate.
- Real delta_x () const

    Provides access to the computed $ x $.
- const Real ∗ discrete_domain_x () const

    Provides access to the grid spatial data.
- Real ∗ discrete_field ()

    Provides access to the grid field data.
- int num_cells_x () const

    Provides access to the number of cells of the grid.
- void BindScalarField (Real(∗ScalarField)(const Real &xx))

    Binds a given scalar field to the grid.
- void BindVectorField (Real(∗VectorField)(Real xx))

    Binds a given vector field to the grid.
- bool WriteToFile (std::string filename, std::string space_name, std::string field_name) const

    Writes grid to a file compatible with gnuplot 4.6.

Private Attributes

- FieldNature nature_

    Nature of the discrete field.
- std::vector< Real > discrete_domain_x_

    Array of spatial data.
- std::vector< Real > discrete_field_

    Array of field's data.
- Real west_bndy_x_

    West boundary spatial coordinate.
- Real east_bndy_x_

    East boundary spatial coordinate.
- Real num_cells_x_

    Number of cells discretizing the domain.
- Real delta_x_

    Produced Δx.

Friends

- std::ostream & operator$<<$ (std::ostream &stream, UniStgGrid1D &in)

    Prints the grid as a tuple of arrays.

16.23.1 Detailed Description

Uniform 1D Staggered Grid.

Definition at line 77 of file mtk_uni_stg_grid_1d.h.

16.23.2 Constructor & Destructor Documentation

16.23.2.1 mtk::UniStgGrid1D::UniStgGrid1D ( )

Definition at line 99 of file mtk_uni_stg_grid_1d.cc.

16.23.2.2 mtk::UniStgGrid1D::UniStgGrid1D (const UniStgGrid1D & *grid*)

Parameters

in	*grid*	Given grid.

Definition at line 108 of file mtk_uni_stg_grid_1d.cc.

16.23.2.3 mtk::UniStgGrid1D::UniStgGrid1D (const Real & *west_bndy_x,* const Real & *east_bndy_x,* const int & *num_cells_x,* const mtk::FieldNature & *nature* = mtk::SCALAR)

Parameters

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

See also

> mtk::FieldNature

Definition at line 124 of file mtk_uni_stg_grid_1d.cc.

Here is the call graph for this function:

16.23.2.4   mtk::UniStgGrid1D::∼UniStgGrid1D (   )

Definition at line 144 of file mtk_uni_stg_grid_1d.cc.

16.23.3   Member Function Documentation

16.23.3.1   void mtk::UniStgGrid1D::BindScalarField (Real(∗)(const Real &xx) *ScalarField*)

Parameters

in	*ScalarField*	Pointer to the function implementing the scalar field.

1. Create collection of spatial coordinates.

2. Create collection of field samples.

Definition at line 176 of file mtk_uni_stg_grid_1d.cc.

Here is the call graph for this function:
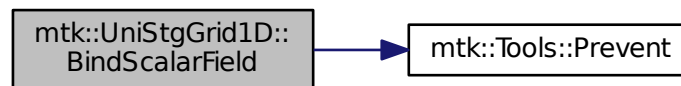
16.23.3.2   void mtk::UniStgGrid1D::BindVectorField (Real(∗)(Real xx) *VectorField*)

We assume the field to be of the form:

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

Parameters

in	*VectorField*	Pointer to the function implementing the vector field.

1. Create collection of spatial coordinates.

2. Create collection of field samples.

Definition at line 212 of file mtk_uni_stg_grid_1d.cc.

Here is the call graph for this function:
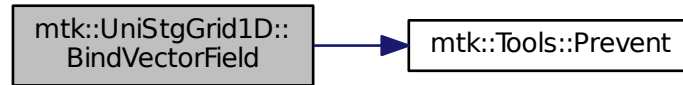
16.23.3.3 mtk::Real mtk::UniStgGrid1D::delta_x ( ) const

Returns

Computed $ x $.

Definition at line 156 of file mtk_uni_stg_grid_1d.cc.

Here is the caller graph for this function:

16.23.3.4 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.

16.23.3.5 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:
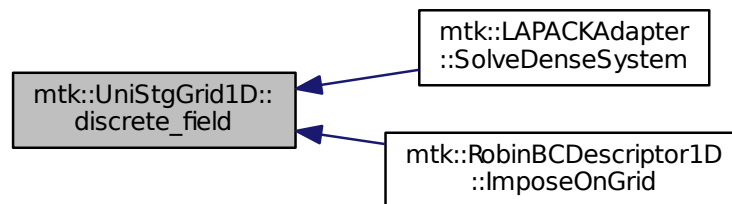
16.23.3.6 mtk::Real mtk::UniStgGrid1D::east_bndy_x ( ) const

Returns

East boundary spatial coordinate.

Definition at line 151 of file mtk_uni_stg_grid_1d.cc.

16.23.3.7 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:

16.23.3.8 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.

16.23.3.9 bool mtk::UniStgGrid1D::WriteToFile (std::string *filename,* std::string *space_name,* std::string *field_name*) const

Parameters

in	*filename*	Name of the output file.
in	*space_name*	Name for the first column of the data.
in	*field_name*	Name for the second column of the data.

Returns

    Success of the file writing process.

See also

    http://www.gnuplot.info/

Definition at line 240 of file mtk_uni_stg_grid_1d.cc.

16.23.4 Friends And Related Function Documentation

16.23.4.1 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.

16.23.5 Member Data Documentation

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

Definition at line 199 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 193 of file mtk_uni_stg_grid_1d.h.

16.23.5.3 std::vector$<$**Real**$>$ **mtk::UniStgGrid1D::discrete_field_** `[private]`

Definition at line 194 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 197 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 191 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 198 of file mtk_uni_stg_grid_1d.h.

16.23.5.7 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

16.24 mtk::UniStgGrid2D Class Reference

Uniform 2D Staggered Grid.

```
#include <mtk_uni_stg_grid_2d.h>
```

Collaboration diagram for mtk::UniStgGrid2D:
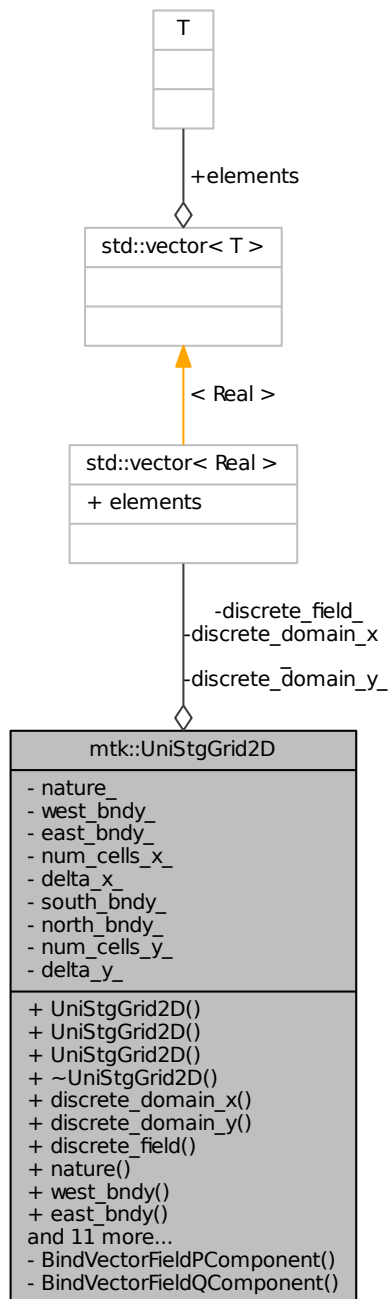
Public Member Functions

- UniStgGrid2D ()

Default constructor.

- UniStgGrid2D (const UniStgGrid2D &grid)

    Copy constructor.

- UniStgGrid2D (const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const Real &south_bndy_y, const Real &north_bndy_y, const int &num_cells_y, const mtk::FieldNature &nature=mtk::S↩ CALAR)

    Construct a grid based on spatial discretization parameters.

- ∼UniStgGrid2D ()

    Destructor.

- const Real ∗ discrete_domain_x () const

    Provides access to the grid spatial data.

- const Real ∗ discrete_domain_y () const

    Provides access to the grid spatial data.

- Real ∗ discrete_field ()

    Provides access to the grid field data.

- FieldNature nature () const

    Physical nature of the data bound to the grid.

- Real west_bndy () const

    Provides access to west boundary spatial coordinate.

- Real east_bndy () const

    Provides access to east boundary spatial coordinate.

- int num_cells_x () const

    Provides access to the number of cells of the grid.

- Real delta_x () const

    Provides access to the computed $ x $.

- Real south_bndy () const

    Provides access to south boundary spatial coordinate.

- Real north_bndy () const

    Provides access to north boundary spatial coordinate.

- int num_cells_y () const

    Provides access to the number of cells of the grid.

- Real delta_y () const

    Provides access to the computed $ y $.

- bool Bound () const

    Have any field been bound to the grid?

- int Size () const

    Total number of samples in the grid.

- void BindScalarField (Real(∗ScalarField)(const Real &xx, const Real &yy))

    Binds a given scalar field to the grid.

- void BindVectorField (Real(∗VectorFieldPComponent)(const Real &xx, const Real &yy), Real(∗VectorFieldQ↩ Component)(const Real &xx, const Real &yy))

    Binds a given vector field to the grid.

- bool WriteToFile (std::string filename, std::string space_name_x, std::string space_name_y, std::string field_↩ name) const

    Writes grid to a file compatible with Gnuplot 4.6.

Private Member Functions

- void BindVectorFieldPComponent (Real(∗VectorFieldPComponent)(const Real &xx, const Real &yy))

  Binds a given component of a vector field to the grid.
- void BindVectorFieldQComponent (Real(∗VectorFieldQComponent)(const Real &xx, const Real &yy))

  Binds a given component of a vector field to the grid.

Private Attributes

- std::vector< Real > discrete_domain_x_

  Array of spatial data.
- std::vector< Real > discrete_domain_y_

  Array of spatial data.
- std::vector< Real > discrete_field_

  Array of field's data.
- FieldNature nature_

  Nature of the discrete field.
- Real west_bndy_

  West boundary spatial coordinate.
- Real east_bndy_

  East boundary spatial coordinate.
- int num_cells_x_

  Number of cells discretizing the domain.
- Real delta_x_

  Computed Δx.
- Real south_bndy_

  West boundary spatial coordinate.
- Real north_bndy_

  East boundary spatial coordinate.
- int num_cells_y_

  Number of cells discretizing the domain.
- Real delta_y_

  Computed Δy.

Friends

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

  Prints the grid as a tuple of arrays.

16.24.1 Detailed Description

Uniform 2D Staggered Grid.

Definition at line 79 of file mtk_uni_stg_grid_2d.h.

16.24.2 Constructor & Destructor Documentation

16.24.2.1 mtk::UniStgGrid2D::UniStgGrid2D ( )

Definition at line 131 of file mtk_uni_stg_grid_2d.cc.

16.24.2.2 mtk::UniStgGrid2D::UniStgGrid2D (const UniStgGrid2D & *grid*)

Parameters

in	*grid*	Given grid.

Definition at line 145 of file mtk_uni_stg_grid_2d.cc.

16.24.2.3 mtk::UniStgGrid2D::UniStgGrid2D (const Real & *west_bndy_x,* const Real & *east_bndy_x,* const int & *num_cells_x,* const Real & *south_bndy_y,* const Real & *north_bndy_y,* const int & *num_cells_y,* const mtk::FieldNature & *nature =* mtk::SCALAR)

Parameters

in	*west_bndy_x*	Coordinate for the west boundary.
in	*east_bndy_x*	Coordinate for the east boundary.
in	*num_cells_x*	Number of cells of the required grid.
in	*south_bndy_y*	Coordinate for the west boundary.
in	*north_bndy_y*	Coordinate for the east boundary.
in	*num_cells_y*	Number of cells of the required grid.
in	*nature*	Nature of the discrete field to hold.

See also

> mtk::FieldNature

Definition at line 169 of file mtk_uni_stg_grid_2d.cc.

Here is the call graph for this function:

16.24.2.4 mtk::UniStgGrid2D::∼UniStgGrid2D ( )

Definition at line 203 of file mtk_uni_stg_grid_2d.cc.

16.24.3 Member Function Documentation

16.24.3.1 void mtk::UniStgGrid2D::BindScalarField (Real(∗)(const Real &xx, const Real &yy) *ScalarField*)

Parameters

in	*ScalarField*	Pointer to the function implementing the scalar field.

1. Create collection of spatial coordinates for x.

2. Create collection of spatial coordinates for y.

3. Create collection of field samples.

Definition at line 275 of file mtk_uni_stg_grid_2d.cc.

Here is the call graph for this function:

16.24.3.2 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	*VectorFieldP↩ Component*	Pointer to the function implementing the $ p $ component of the vector field.
in	*VectorFieldP↩ Component*	Pointer to the function implementing the $ q $ component of the vector field.

Definition at line 423 of file mtk_uni_stg_grid_2d.cc.

Here is the call graph for this function:

16.24.3.3   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	*BindVectorField↩ PComponent*	Pointer to the function implementing the p component of the vector field.

1. Create collection of spatial coordinates for x.

2. Create collection of spatial coordinates for y.

3. Allocate space for discrete vector field and bind p component.

Definition at line 330 of file mtk_uni_stg_grid_2d.cc.

16.24.3.4   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	*BindVectorField↩ QComponent*	Pointer to the function implementing the q component of the vector field.

1. Bind q component, since p component has already been bound.

Definition at line 395 of file mtk_uni_stg_grid_2d.cc.

16.24.3.5   bool mtk::UniStgGrid2D::Bound (   ) const

Returns

> True is a field has been bound.

Definition at line 255 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:

```
┌──────────────────────┐          ┌────────────────────────────┐
│ mtk::UniStgGrid2D::   │◄─────────│ mtk::RobinBCDescriptor2D    │
│        Bound          │          │ ::ImposeOnLaplacianMatrix   │
└──────────────────────┘          └────────────────────────────┘
```

16.24.3.6   mtk::Real mtk::UniStgGrid2D::delta_x ( ) const

Returns

> Computed x.

Definition at line 225 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:

```
                                  ┌────────────────────────────────┐
                                  │ mtk::RobinBCDescriptor2D        │
                                  │ ::ImposeOnSouthBoundaryNoSpace  │
                                  └────────────────────────────────┘
┌──────────────────────┐          ┌────────────────────────────────┐
│ mtk::UniStgGrid2D::   │◄─────────│ mtk::RobinBCDescriptor2D        │
│        delta_x        │          │ ::ImposeOnNorthBoundaryNoSpace  │
└──────────────────────┘          └────────────────────────────────┘
                                  ┌────────────────────────────┐
                                  │ mtk::RobinBCDescriptor2D    │
                                  │ ::ImposeOnGrid              │
                                  └────────────────────────────┘
```
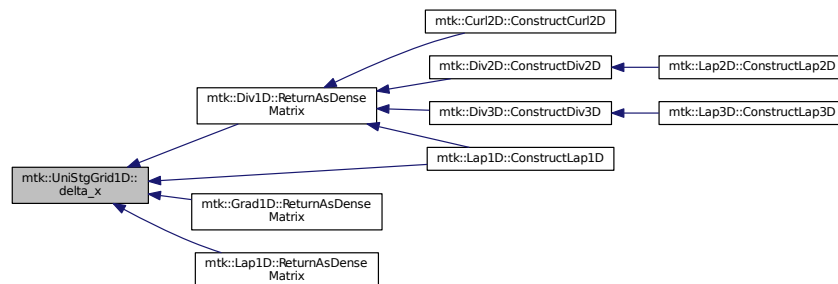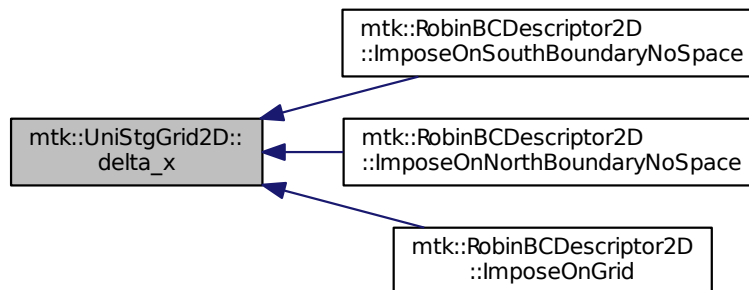
16.24.3.7   mtk::Real mtk::UniStgGrid2D::delta_y ( ) const

Returns

Computed $ y $.

Definition at line 250 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
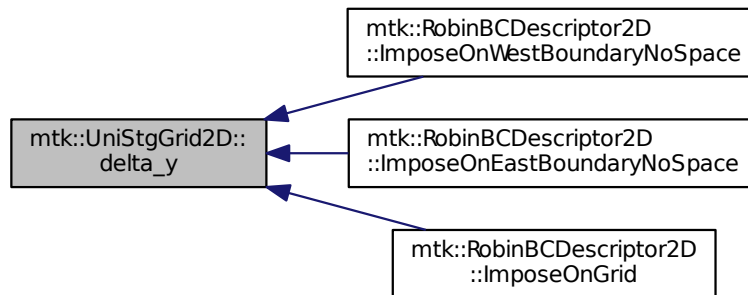
16.24.3.8 const mtk::Real ∗ mtk::UniStgGrid2D::discrete_domain_x ( ) const

Returns

Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 230 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
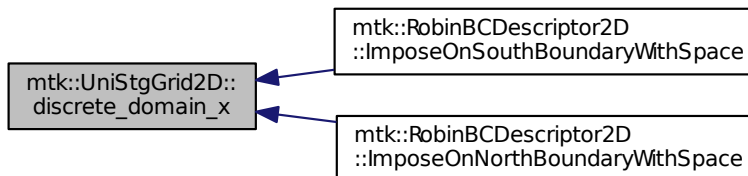
**16.24.3.9 const mtk::Real ∗ mtk::UniStgGrid2D::discrete_domain_y ( ) const**

Returns

>   Pointer to the spatial data.

Todo Review const-correctness of the pointer we return.

Definition at line 260 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
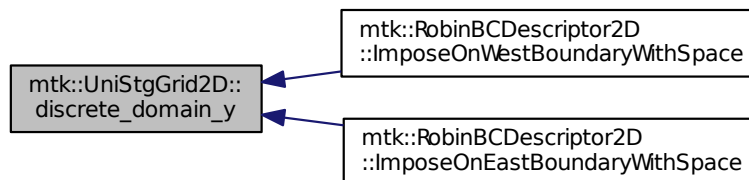
16.24.3.10 **mtk::Real** ∗ **mtk::UniStgGrid2D::discrete_field (** **)**

Returns

>   Pointer to the field data.

Definition at line 265 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
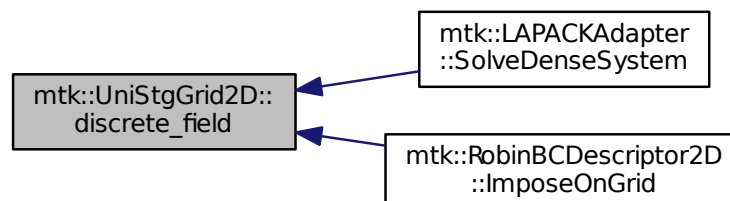
16.24.3.11 **mtk::Real mtk::UniStgGrid2D::east_bndy (** **) const**

Returns

> East boundary spatial coordinate.

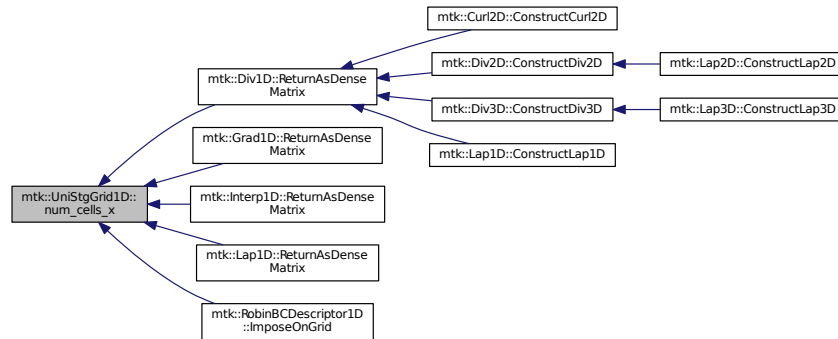Definition at line 215 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
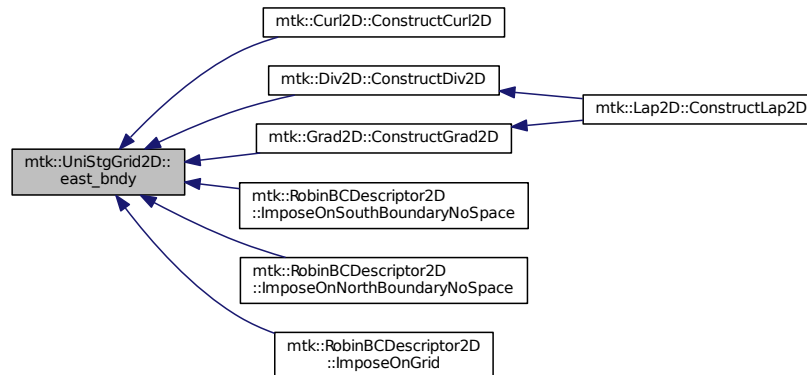
16.24.3.12 mtk::FieldNature mtk::UniStgGrid2D::nature ( ) const

Returns

> Value of an enumeration.

See also

> mtk::FieldNature

Definition at line 205 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
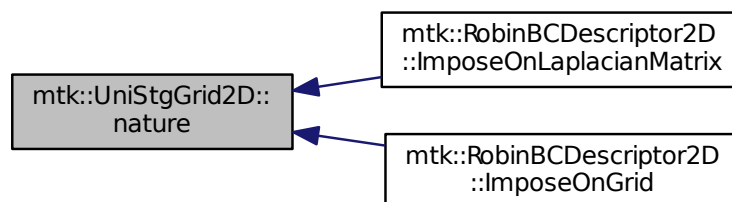
16.24.3.13   mtk::Real mtk::UniStgGrid2D::north_bndy (   ) const

Returns

North boundary spatial coordinate.

Definition at line 240 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
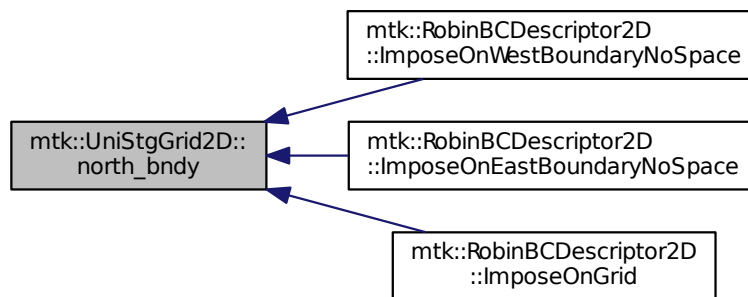
16.24.3.14   int mtk::UniStgGrid2D::num_cells_x (   ) const

Returns

Number of cells of the grid.

Definition at line 220 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
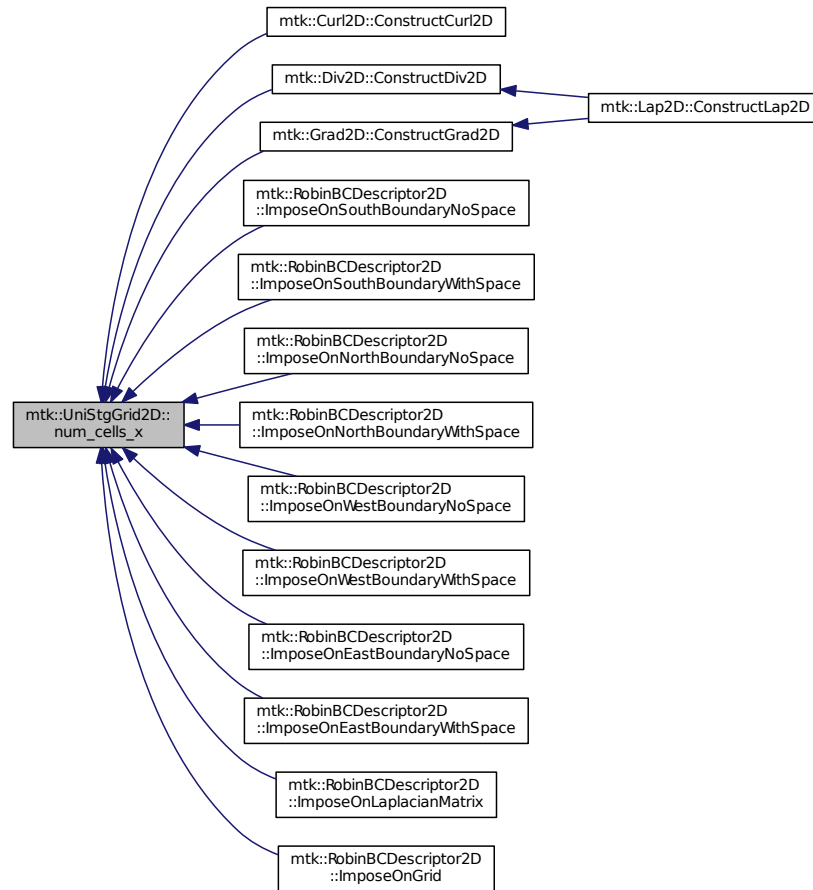
16.24.3.15 int mtk::UniStgGrid2D::num_cells_y ( ) const

Returns

> Number of cells of the grid.

Definition at line 245 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
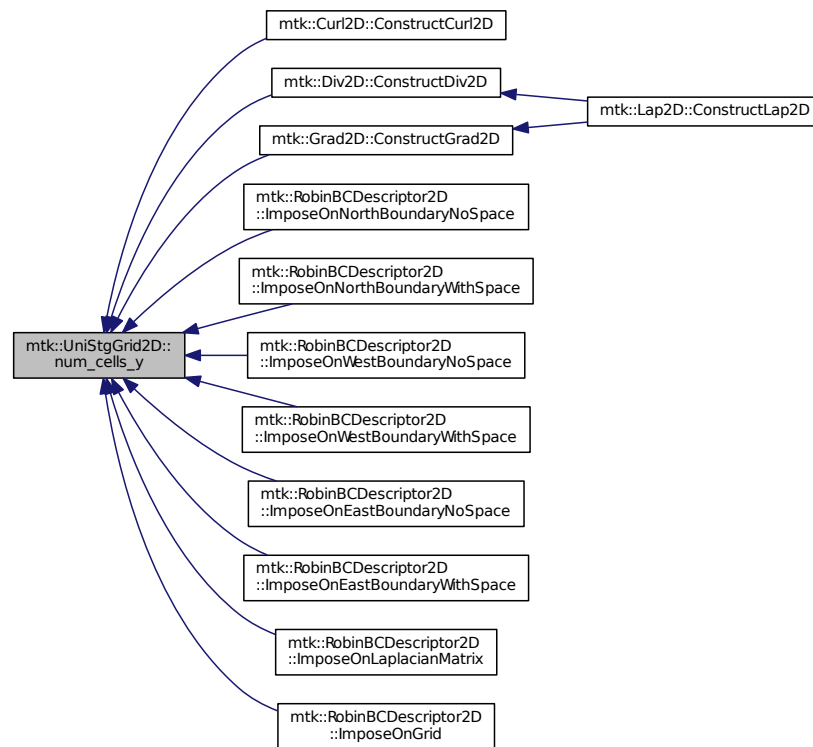
16.24.3.16 int mtk::UniStgGrid2D::Size (   ) const

Returns

> Total number of samples in the grid.

Definition at line 270 of file mtk_uni_stg_grid_2d.cc.

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

Returns

>   South boundary spatial coordinate.

Definition at line 235 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
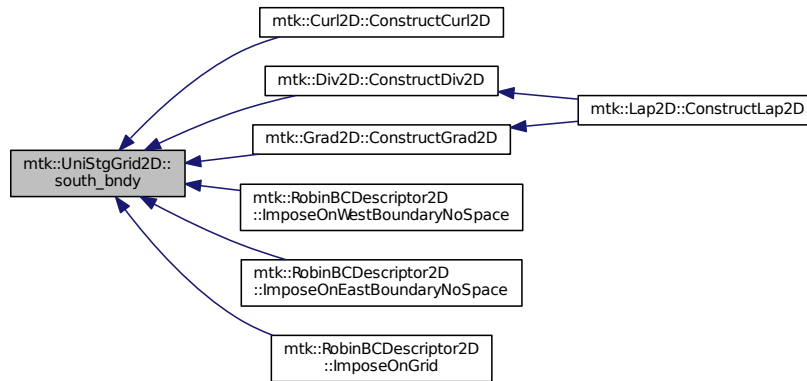
16.24.3.18   mtk::Real mtk::UniStgGrid2D::west_bndy (   ) const

Returns

>   West boundary spatial coordinate.

Definition at line 210 of file mtk_uni_stg_grid_2d.cc.

Here is the caller graph for this function:
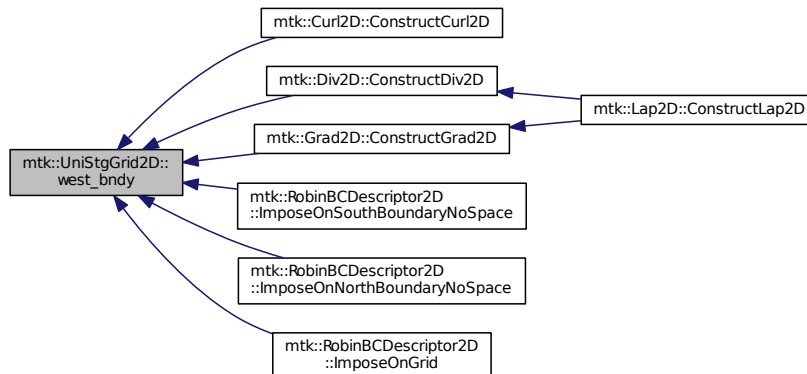
16.24.3.19   bool mtk::UniStgGrid2D::WriteToFile (std::string *filename,* std::string *space_name_x,* std::string *space_name_y,* std::string *field_name*) const

Parameters

in	*filename*	Name of the output file.
in	*space_name_x*	Name for the first column of the (spatial) data.
in	*space_name_y*	Name for the second column of the (spatial) data.
in	*field_name*	Name for the second column of the (physical field) data.

Returns

> Success of the file writing process.

See also

> [`http://www.gnuplot.info/`](http://www.gnuplot.info/)

Write the values of the p component, with a null q component.

Write the values of the q component, with a null p component.

Definition at line 435 of file mtk_uni_stg_grid_2d.cc.

16.24.4   Friends And Related Function Documentation

16.24.4.1   std::ostream& operator$\ll$**(std::ostream &** *stream,* **mtk::UniStgGrid2D &** *in* **)**   `[friend]`

1. Print spatial coordinates.

2. Print scalar field.

Definition at line 67 of file mtk_uni_stg_grid_2d.cc.

16.24.5   Member Data Documentation

16.24.5.1   Real mtk::UniStgGrid2D::delta_x_   `[private]`

Definition at line 302 of file mtk_uni_stg_grid_2d.h.

16.24.5.2   Real mtk::UniStgGrid2D::delta_y_   `[private]`

Definition at line 307 of file mtk_uni_stg_grid_2d.h.

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

Definition at line 293 of file mtk_uni_stg_grid_2d.h.

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

Definition at line 294 of file mtk_uni_stg_grid_2d.h.

16.24.5.5 std::vector<Real> mtk::UniStgGrid2D::discrete_field_ `[private]`

Definition at line 295 of file mtk_uni_stg_grid_2d.h.

16.24.5.6 Real mtk::UniStgGrid2D::east_bndy_ `[private]`

Definition at line 300 of file mtk_uni_stg_grid_2d.h.

16.24.5.7 FieldNature mtk::UniStgGrid2D::nature_ `[private]`

Definition at line 297 of file mtk_uni_stg_grid_2d.h.

16.24.5.8 Real mtk::UniStgGrid2D::north_bndy_ `[private]`

Definition at line 305 of file mtk_uni_stg_grid_2d.h.

16.24.5.9 int mtk::UniStgGrid2D::num_cells_x_ `[private]`

Definition at line 301 of file mtk_uni_stg_grid_2d.h.

16.24.5.10 int mtk::UniStgGrid2D::num_cells_y_ `[private]`

Definition at line 306 of file mtk_uni_stg_grid_2d.h.

16.24.5.11 Real mtk::UniStgGrid2D::south_bndy_ `[private]`

Definition at line 304 of file mtk_uni_stg_grid_2d.h.

16.24.5.12 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

16.25 mtk::UniStgGrid3D Class Reference

Uniform 3D Staggered Grid.

```
#include <mtk_uni_stg_grid_3d.h>
```

Collaboration diagram for mtk::UniStgGrid3D:

```
                          ┌─────────────────┐
                          │        T        │
                          ├─────────────────┤
                          │                 │
                          ├─────────────────┤
                          │                 │
                          └─────────────────┘
                                   │
                                   │ +elements
                                   ◇
                          ┌─────────────────┐
                          │ std::vector< T >│
                          ├─────────────────┤
                          │                 │
                          ├─────────────────┤
                          │                 │
                          └─────────────────┘
                                   ▲
                                   │ < Real >
                          ┌──────────────────────┐
                          │ std::vector< Real >  │
                          ├──────────────────────┤
                          │ + elements           │
                          ├──────────────────────┤
                          │                      │
                          └──────────────────────┘
                                   │
                                   │ -discrete_domain_z_
                                   │ -discrete_field_
                                   │ -discrete_domain
                                   │        _x_
                                   │ -discrete_domain_y_
                                   ◇
```

mtk::UniStgGrid3D
- nature_   - west_bndy_   - east_bndy_   - num_cells_x_   - delta_x_   - south_bndy_   - north_bndy_   - num_cells_y_   - delta_y_   - bottom_bndy_   - top_bndy_   - num_cells_z_   - delta_z_
+ operator=()   + UniStgGrid3D()   + UniStgGrid3D()   + UniStgGrid3D()   + ~UniStgGrid3D()   + discrete_domain_x()   + discrete_domain_y()   + discrete_domain_z()   + discrete_field()   + nature()   and 17 more...   - BindVectorFieldPComponent()   - BindVectorFieldQComponent()   - BindVectorFieldRComponent()

Public Member Functions

- UniStgGrid3D operator= (const UniStgGrid3D &in)

Overloaded assignment operator.

- UniStgGrid3D ()

    Default constructor.

- UniStgGrid3D (const UniStgGrid3D &grid)

    Copy constructor.

- UniStgGrid3D (const Real &west_bndy_x, const Real &east_bndy_x, const int &num_cells_x, const Real &south_bndy_y, const Real &north_bndy_y, const int &num_cells_y, const Real &bottom_bndy_z, const Real &top_bndy_z, const int &num_cells_z, const mtk::FieldNature &nature=mtk::SCALAR)

    Construct a grid based on spatial discretization parameters.

- ∼UniStgGrid3D ()

    Destructor.

- const Real ∗ discrete_domain_x () const

    Provides access to the grid spatial data.

- const Real ∗ discrete_domain_y () const

    Provides access to the grid spatial data.

- const Real ∗ discrete_domain_z () const

    Provides access to the grid spatial data.

- Real ∗ discrete_field ()

    Provides access to the grid field data.

- FieldNature nature () const

    Physical nature of the data bound to the grid.

- Real west_bndy () const

    Provides access to west boundary spatial coordinate.

- Real east_bndy () const

    Provides access to east boundary spatial coordinate.

- int num_cells_x () const

    Provides access to the number of cells of the grid.

- Real delta_x () const

    Provides access to the computed $ x $.

- Real south_bndy () const

    Provides access to south boundary spatial coordinate.

- Real north_bndy () const

    Provides access to north boundary spatial coordinate.

- int num_cells_y () const

    Provides access to the number of cells of the grid.

- Real delta_y () const

    Provides access to the computed $ y $.

- Real bottom_bndy () const

    Provides access to bottom boundary spatial coordinate.

- Real top_bndy () const

    Provides access to top boundary spatial coordinate.

- int num_cells_z () const

    Provides access to the number of cells of the grid.

- Real delta_z () const

    Provides access to the computed $ z $.

- bool Bound () const

    Have any field been bound to the grid?

- int Size () const

    Total number of samples in the grid.
- void BindScalarField (Real(∗ScalarField)(const Real &xx, const Real &yy, const Real &zz))

    Binds a given scalar field to the grid.
- void BindVectorField (Real(∗VectorFieldPComponent)(const Real &xx, const Real &yy, const Real &zz), Real(∗VectorFieldQComponent)(const Real &xx, const Real &yy, const Real &zz), Real(∗VectorFieldR↩Component)(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.

16.25.1 Detailed Description

Uniform 3D Staggered Grid.

Definition at line 79 of file mtk_uni_stg_grid_3d.h.

16.25.2 Constructor & Destructor Documentation

16.25.2.1 mtk::UniStgGrid3D::UniStgGrid3D ( )

Definition at line 123 of file mtk_uni_stg_grid_3d.cc.

16.25.2.2 mtk::UniStgGrid3D::UniStgGrid3D (const UniStgGrid3D & *grid*)

Parameters

in	*grid*	Given grid.

Definition at line 142 of file mtk_uni_stg_grid_3d.cc.

16.25.2.3 mtk::UniStgGrid3D::UniStgGrid3D (const **Real** & *west_bndy_x,* const **Real** & *east_bndy_x,* const int & *num_cells_x,* const **Real** & *south_bndy_y,* const **Real** & *north_bndy_y,* const int & *num_cells_y,* const **Real** & *bottom_bndy_z,* const **Real** & *top_bndy_z,* const int & *num_cells_z,* const **mtk::FieldNature** & *nature =* **mtk::SCALAR**)

Parameters

in	*west_bndy_x*	Coordinate for the west boundary.
in	*east_bndy_x*	Coordinate for the east boundary.
in	*num_cells_x*	Number of cells of the required grid.
in	*south_bndy_y*	Coordinate for the west boundary.
in	*north_bndy_y*	Coordinate for the east boundary.
in	*num_cells_y*	Number of cells of the required grid.
in	*bottom_bndy_z*	Coordinate for the bottom boundary.
in	*top_bndy_z*	Coordinate for the top boundary.
in	*num_cells_z*	Number of cells of the required grid.
in	*nature*	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:

**16.25.2.4 mtk::UniStgGrid3D::∼UniStgGrid3D ( )**

Definition at line 221 of file mtk_uni_stg_grid_3d.cc.

16.25.3 Member Function Documentation

16.25.3.1 void mtk::UniStgGrid3D::BindScalarField (Real(∗)(const Real &xx, const Real &yy, const Real &zz) *ScalarField*)

Parameters

in	*ScalarField*	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:

```
mtk::UniStgGrid3D::        mtk::Tools::Prevent
BindScalarField
```
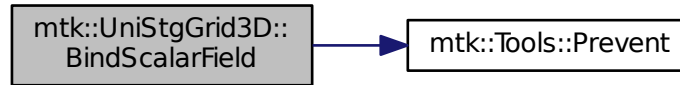
16.25.3.2   void mtk::UniStgGrid3D::BindVectorField (Real(∗)(const Real &xx, const Real &yy, const Real &zz)
         VectorFieldPComponent, **Real(∗)(const Real &xx, const Real &yy, const Real &zz)** *VectorFieldQComponent,*
         Real(∗)(const Real &xx, const Real &yy, const Real &zz) *VectorFieldRComponent)*

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	*VectorFieldP↩ Component*	Pointer to the function implementing the $ p $ component of the vector field.
in	*VectorFieldP↩ Component*	Pointer to the function implementing the $ q $ component of the vector field.
in	*VectorFieldR↩ Component*	Pointer to the function implementing the $ r $ component of the vector field.

Definition at line 414 of file mtk_uni_stg_grid_3d.cc.

Here is the call graph for this function:

```
mtk::UniStgGrid3D::        mtk::Tools::Prevent
BindVectorField
```

16.25.3.3   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	*BindVectorField↩* *PComponent*	Pointer to the function implementing the p component of the vector field.

Definition at line 393 of file mtk_uni_stg_grid_3d.cc.

16.25.3.4 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	*BindVectorField↩* *QComponent*	Pointer to the function implementing the q component of the vector field.

Definition at line 400 of file mtk_uni_stg_grid_3d.cc.

16.25.3.5 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	*BindVectorField↩* *QComponent*	Pointer to the function implementing the r component of the vector field.

Definition at line 407 of file mtk_uni_stg_grid_3d.cc.

16.25.3.6 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:
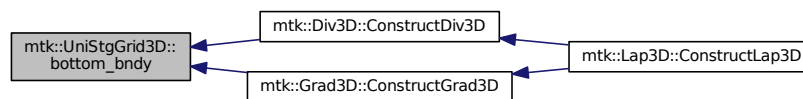
16.25.3.7   bool mtk::UniStgGrid3D::Bound (   ) const

Returns

True is a field has been bound.

Definition at line 308 of file mtk_uni_stg_grid_3d.cc.

16.25.3.8   mtk::Real mtk::UniStgGrid3D::delta_x (   ) const

Returns

Computed $ x $.

Definition at line 243 of file mtk_uni_stg_grid_3d.cc.

16.25.3.9   mtk::Real mtk::UniStgGrid3D::delta_y (   ) const

Returns

Computed $ y $.

Definition at line 268 of file mtk_uni_stg_grid_3d.cc.

16.25.3.10   mtk::Real mtk::UniStgGrid3D::delta_z (   ) const

Returns

Computed $ z $.

Definition at line 293 of file mtk_uni_stg_grid_3d.cc.

16.25.3.11   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.

16.25.3.12   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.

16.25.3.13 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.

16.25.3.14 mtk::Real ∗ mtk::UniStgGrid3D::discrete_field ( )

Returns

Pointer to the field data.

Definition at line 303 of file mtk_uni_stg_grid_3d.cc.

16.25.3.15 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:
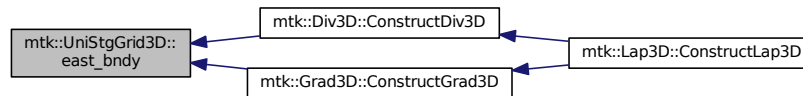
16.25.3.16 mtk::FieldNature mtk::UniStgGrid3D::nature ( ) const

Returns

Value of an enumeration.

See also

mtk::FieldNature

Definition at line 223 of file mtk_uni_stg_grid_3d.cc.

16.25.3.17 mtk::Real mtk::UniStgGrid3D::north_bndy ( ) const

Returns

North boundary spatial coordinate.

Definition at line 258 of file mtk_uni_stg_grid_3d.cc.

16.25.3.18   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:
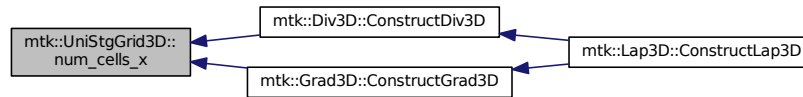
16.25.3.19   int mtk::UniStgGrid3D::num_cells_y ( ) const

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:
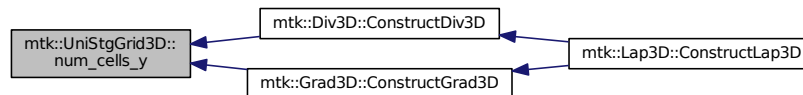
16.25.3.20   int mtk::UniStgGrid3D::num_cells_z ( ) const

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:
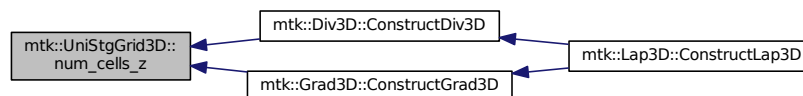
16.25.3.21 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator= (const UniStgGrid3D & *in*)

Parameters

in	*in*	Given grid.

Returns

Copy of the given grid.

Definition at line 116 of file mtk_uni_stg_grid_3d.cc.

16.25.3.22 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.

16.25.3.23 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:
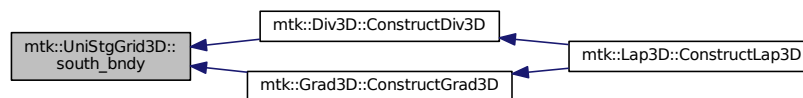
16.25.3.24 mtk::Real mtk::UniStgGrid3D::top_bndy ( ) const

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:
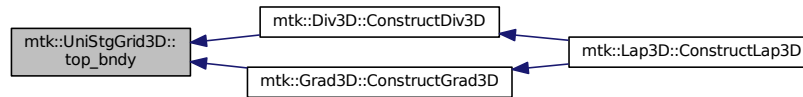
16.25.3.25 mtk::Real mtk::UniStgGrid3D::west_bndy ( ) const

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:
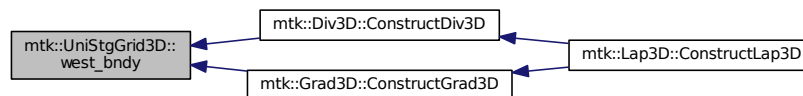
16.25.3.26 bool mtk::UniStgGrid3D::WriteToFile (std::string *filename,* std::string *space_name_x,* std::string *space_name_y,* std::string *space_name_z,* std::string *field_name*) const

Parameters

in	*filename*	Name of the output file.
in	*space_name_x*	Name for the first column of the (spatial) data.
in	*space_name_y*	Name for the second column of the (spatial) data.
in	*space_name_z*	Name for the third column of the (spatial) data.
in	*field_name*	Name for the second column of the (physical field) data.

Returns

Success of the file writing process.

See also

    http://www.gnuplot.info/

Definition at line 433 of file mtk_uni_stg_grid_3d.cc.

16.25.4 Friends And Related Function Documentation

16.25.4.1 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.

16.25.5 Member Data Documentation

16.25.5.1 Real mtk::UniStgGrid3D::bottom_bndy_ `[private]`

Definition at line 396 of file mtk_uni_stg_grid_3d.h.

16.25.5.2 Real mtk::UniStgGrid3D::delta_x_ `[private]`

Definition at line 389 of file mtk_uni_stg_grid_3d.h.

16.25.5.3 Real mtk::UniStgGrid3D::delta_y_ `[private]`

Definition at line 394 of file mtk_uni_stg_grid_3d.h.

16.25.5.4 Real mtk::UniStgGrid3D::delta_z_ `[private]`

Definition at line 399 of file mtk_uni_stg_grid_3d.h.

16.25.5.5 std::vector$<$Real$>$ mtk::UniStgGrid3D::discrete_domain_x_ `[private]`

Definition at line 379 of file mtk_uni_stg_grid_3d.h.

16.25.5.6 std::vector$<$Real$>$ mtk::UniStgGrid3D::discrete_domain_y_ `[private]`

Definition at line 380 of file mtk_uni_stg_grid_3d.h.

16.25.5.7 std::vector$<$Real$>$ mtk::UniStgGrid3D::discrete_domain_z_ `[private]`

Definition at line 381 of file mtk_uni_stg_grid_3d.h.

16.25.5.8   std::vector<**Real**> **mtk::UniStgGrid3D::discrete_field_**   `[private]`

Definition at line 382 of file mtk_uni_stg_grid_3d.h.

16.25.5.9   Real mtk::UniStgGrid3D::east_bndy_   `[private]`

Definition at line 387 of file mtk_uni_stg_grid_3d.h.

16.25.5.10   FieldNature mtk::UniStgGrid3D::nature_   `[private]`

Definition at line 384 of file mtk_uni_stg_grid_3d.h.

16.25.5.11   Real mtk::UniStgGrid3D::north_bndy_   `[private]`

Definition at line 392 of file mtk_uni_stg_grid_3d.h.

16.25.5.12   int mtk::UniStgGrid3D::num_cells_x_   `[private]`

Definition at line 388 of file mtk_uni_stg_grid_3d.h.

16.25.5.13   int mtk::UniStgGrid3D::num_cells_y_   `[private]`

Definition at line 393 of file mtk_uni_stg_grid_3d.h.

16.25.5.14   int mtk::UniStgGrid3D::num_cells_z_   `[private]`

Definition at line 398 of file mtk_uni_stg_grid_3d.h.

16.25.5.15   Real mtk::UniStgGrid3D::south_bndy_   `[private]`

Definition at line 391 of file mtk_uni_stg_grid_3d.h.

16.25.5.16   Real mtk::UniStgGrid3D::top_bndy_   `[private]`

Definition at line 397 of file mtk_uni_stg_grid_3d.h.

16.25.5.17   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 17

File Documentation

17.1 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:
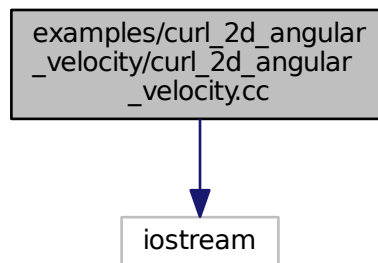
Functions

- int main ()

17.1.1 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.

17.1.2 Function Documentation

17.1.2.1 int main ( )

Definition at line 106 of file curl_2d_angular_velocity.cc.

17.2 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
00084   mtk::Real aa = 0.0;
00085   mtk::Real bb = 4.0;
00086   mtk::Real cc = 0.0;
00087   mtk::Real dd = 4.0;
00088
00089   int nn = 10;
00090   int mm = 10;
00091
00092   mtk::UniStgGrid2D gg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00093
00094   gg.BindVectorField(VectorFieldPComponent, VectorFieldQComponent);
00095
00096   if(!gg.WriteToFile("curl_2d_angular_velocity_gg.dat", "x", "y", "v(x,y)")) {
00097     std::cerr << "Angular field could not be written to disk." << std::endl;
00098     return EXIT_FAILURE;
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
```

17.3 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:
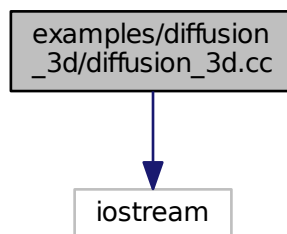
Functions

- int main ()

17.3.1 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.

17.3.2 Function Documentation

17.3.2.1 int main ( )

Definition at line 122 of file diffusion_3d.cc.

17.4 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
00089   mtk::Real west_bndy_x{0.0};
00090   mtk::Real east_bndy_x{1.0};
00091   mtk::Real south_bndy_y{0.0};
00092   mtk::Real north_bndy_y{1.0};
00093   mtk::Real bottom_bndy_z{0.0};
00094   mtk::Real top_bndy_z{1.0};
00095
00096   int num_cells_x{50};
00097   int num_cells_y{50};
00098   int num_cells_z{50};
00099
00100   mtk::UniStgGrid3D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00101                             south_bndy_y, north_bndy_y, num_cells_y,
00102                             bottom_bndy_z, top_bndy_z, num_cells_z);
00103
00105   comp_sol.BindScalarField(InitialCondition);
00106
00107   if(!comp_sol.WriteToFile("diffusion_3d_comp_sol.dat",
00108                      "x",
00109                      "y",
00110                      "z",
00111                      "Initial u(x,y,z)")) {
00112     std::cerr << "Error writing to file." << std::endl;
00113   }
00114
00116 }
00117
00118 #else
00119 #include <iostream>
00120 using std::cout;
00121 using std::endl;
00122 int main () {
00123   cout << "This code HAS to be compiled with support for C++11." << endl;
00124   cout << "Exiting..." << endl;
00125   return EXIT_SUCCESS;
00126 }
00127 #endif
```

17.5 examples/minimalistic_poisson_1d/minimalistic_poisson_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```
Include dependency graph for minimalistic_poisson_1d.cc:
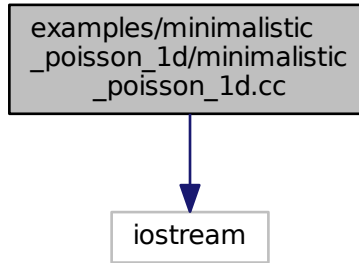
Functions

- int main ()

17.5.1 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:

$$-\breve{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

Finally, we will solve this problem considering $k = 2$.

Author

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

Definition in file minimalistic_poisson_1d.cc.

17.5.2 Function Documentation

17.5.2.1 int main ( )

Definition at line 164 of file minimalistic_poisson_1d.cc.

17.6 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
```

17.7 examples/poisson_1d/poisson_1d.cc File Reference

Poisson Equation on a 1D Uniform Staggered Grid with Robin BCs.

```
#include <iostream>
```
Include dependency graph for poisson_1d.cc:
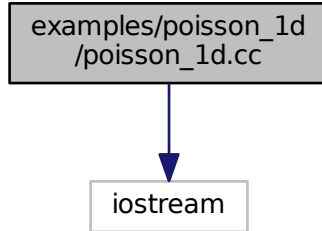
Functions

- int main ()

17.7.1 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:

$$-\breve{\mathbf{L}}_x^k \tilde{p} = \tilde{s}.$$

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.

17.7.2 Function Documentation

17.7.2.1 int main ( )

Definition at line 263 of file poisson_1d.cc.

17.8 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
00143   mtk::Real west_bndy_x{0.0};
00144   mtk::Real east_bndy_x{1.0};
00145   int num_cells_x{5};
00146
00147   mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00148
00150   mtk::Lap1D lap;
00151
00152   if (!lap.ConstructLap1D()) {
00153     std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00154     return EXIT_FAILURE;
00155   }
00156
00157   std::cout << "lap=" << std::endl;
00158   std::cout << lap << std::endl;
00159
00160   mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00161
00162   std::cout << "lapm =" << std::endl;
00163   std::cout << lapm << std::endl;
00164
00166
00167   lapm = mtk::BLASAdapter::RealDenseSM(-1.0, lapm);
00168
00169   std::cout << "-lapm =" << std::endl;
00170   std::cout << lapm << std::endl;
00171
00173   mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00174
00175   source.BindScalarField(Source);
00176
00177   std::cout << "source =" << std::endl;
00178   std::cout << source << std::endl;
00179
00181   mtk::RobinBCDescriptor1D robin_bc_desc_1d;
00182
00183   robin_bc_desc_1d.PushBackWestCoeff(Alpha);
00184   robin_bc_desc_1d.PushBackWestCoeff(Beta);
00185
00186   robin_bc_desc_1d.PushBackEastCoeff(Alpha);
00187   robin_bc_desc_1d.PushBackEastCoeff(Beta);
00188
00189   robin_bc_desc_1d.set_west_condition(Omega);
00190   robin_bc_desc_1d.set_east_condition(Epsilon);
00191
00192   if (!robin_bc_desc_1d.ImposeOnLaplacianMatrix(lap, lapm)) {
00193     std::cerr << "BCs  could not be bound to the matrix." << std::endl;
00194     return EXIT_FAILURE;
```

```
00195   }
00196
00197   std::cout << "Mimetic Laplacian operator with imposed BCs:" << std::endl;
00198   std::cout << lapm << std::endl;
00199
00200   if (!lapm.WriteToFile("poisson_1d_lapm.dat")) {
00201     std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00202     return EXIT_FAILURE;
00203   }
00204
00206   robin_bc_desc_1d.ImposeOnGrid(source);
00207
00208   std::cout << "source =" << std::endl;
00209   std::cout << source << std::endl;
00210
00211   if (!source.WriteToFile("poisson_1d_source.dat", "x", "s(x)")) {
00212     std::cerr << "Source term could not be written to disk." << std::endl;
00213     return EXIT_FAILURE;
00214   }
00215
00217   int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00218
00219   if (!info) {
00220     std::cout << "System solved." << std::endl;
00221     std::cout << std::endl;
00222   } else {
00223     std::cerr << "Something wrong solving system! info = " << info << std::endl;
00224     std::cerr << "Exiting..." << std::endl;
00225     return EXIT_FAILURE;
00226   }
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 = ";
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
```

17.9 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:
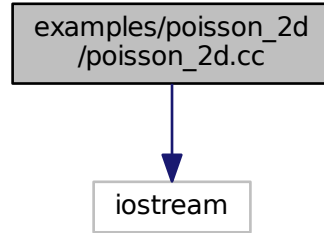
Functions

- int main ()

17.9.1 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.

17.9.2 Function Documentation

17.9.2.1 int main ( )

Definition at line 241 of file poisson_2d.cc.

17.10 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
00144   mtk::Real west_bndy_x{0.0};
00145   mtk::Real east_bndy_x{1.0};
00146   mtk::Real south_bndy_y{0.0};
00147   mtk::Real north_bndy_y{1.0};
00148   int num_cells_x{5};
00149   int num_cells_y{5};
00150
00151   mtk::UniStgGrid2D comp_sol(west_bndy_x, east_bndy_x, num_cells_x,
00152                             south_bndy_y, north_bndy_y, num_cells_y);
00153
00155   mtk::Lap2D lap;
00156
00157   if (!lap.ConstructLap2D(comp_sol)) {
00158     std::cerr << "Mimetic Laplacian could not be built." << std::endl;
00159     return EXIT_FAILURE;
00160   }
00161
00162   mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix());
00163
00165   mtk::UniStgGrid2D source(west_bndy_x, east_bndy_x, num_cells_x,
00166                             south_bndy_y, north_bndy_y, num_cells_y);
00167
00168   source.BindScalarField(Source);
00169
00171   mtk::RobinBCDescriptor2D bcd;
00172
00173   bcd.PushBackWestCoeff(BCCoeff);
00174   bcd.PushBackEastCoeff(BCCoeff);
00175   bcd.PushBackSouthCoeff(BCCoeff);
00176   bcd.PushBackNorthCoeff(BCCoeff);
00177
00178   bcd.ImposeOnLaplacianMatrix(lap, comp_sol, lapm);
00179
00180   if (!lapm.WriteToFile("poisson_2d_lapm.dat")) {
00181     std::cerr << "Laplacian matrix could not be written to disk." << std::endl;
00182     return EXIT_FAILURE;
00183   }
00184
00186   bcd.set_west_condition(WestBC);
00187   bcd.set_east_condition(EastBC);
00188   bcd.set_south_condition(SouthBC);
00189   bcd.set_north_condition(NorthBC);
00190
```

```
00191    bcd.ImposeOnGrid(source);
00192
00193    if(!source.WriteToFile("poisson_2d_source.dat", "x", "y", "s(x,y)")) {
00194      std::cerr << "Source term could not be written to disk." << std::endl;
00195      return EXIT_FAILURE;
00196    }
00197
00199    int info{mtk::LAPACKAdapter::SolveDenseSystem(lapm, source)};
00200
00201    if (!info) {
00202      std::cout << "System solved." << std::endl;
00203      std::cout << std::endl;
00204    } else {
00205      std::cerr << "Something wrong solving system! info = " << info << std::endl;
00206      std::cerr << "Exiting..." << std::endl;
00207      return EXIT_FAILURE;
00208    }
00209
00210    if (!source.WriteToFile("poisson_2d_comp_sol.dat", "x", "y", "~u(x,y)")) {
00211      std::cerr << "Solution could not be written to file." << std::endl;
00212      return EXIT_FAILURE;
00213    }
00214
00216    mtk::UniStgGrid2D known_sol(west_bndy_x, east_bndy_x, num_cells_x,
00217                               south_bndy_y, north_bndy_y, num_cells_y);
00218
00219    known_sol.BindScalarField(KnownSolution);
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
```

17.11   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:
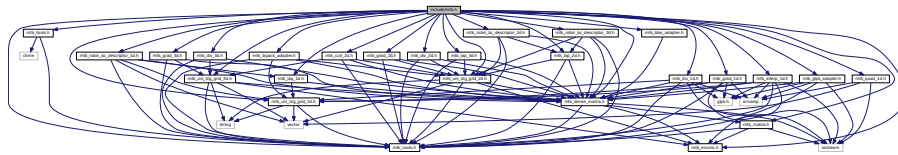
17.11.1 Detailed Description

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

Warning

It is extremely important that the headers are added to this file in a specific order; that is, considering the dependence between the classes these contain.

Author

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

Definition in file mtk.h.

17.12 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 */
00381 #ifndef MTK_INCLUDE_MTK_H_
00382 #define MTK_INCLUDE_MTK_H_
00383
00391 #include "mtk_roots.h"
00392
00400 #include "mtk_enums.h"
00401
00409 #include "mtk_tools.h"
00410
00418 #include "mtk_matrix.h"
00419 #include "mtk_dense_matrix.h"
00420
00428 #include "mtk_blas_adapter.h"
00429 #include "mtk_lapack_adapter.h"
00430 #include "mtk_glpk_adapter.h"
00431
00439 #include "mtk_uni_stg_grid_1d.h"
00440 #include "mtk_uni_stg_grid_2d.h"
00441 #include "mtk_uni_stg_grid_3d.h"
00442
00450 #include "mtk_grad_1d.h"
00451 #include "mtk_div_1d.h"
00452 #include "mtk_lap_1d.h"
00453 #include "mtk_robin_bc_descriptor_1d.h"
00454 #include "mtk_quad_1d.h"
00455 #include "mtk_interp_1d.h"
00456
00457 #include "mtk_grad_2d.h"
00458 #include "mtk_div_2d.h"
00459 #include "mtk_curl_2d.h"
00460 #include "mtk_lap_2d.h"
00461 #include "mtk_robin_bc_descriptor_2d.h"
```

```
00462
00463 #include "mtk_grad_3d.h"
00464 #include "mtk_div_3d.h"
00465 #include "mtk_lap_3d.h"
00466 #include "mtk_robin_bc_descriptor_3d.h"
00467
00468 #endif // End of: MTK_INCLUDE_MTK_H_
```

17.13 include/mtk_blas_adapter.h File Reference

Adapter class for the BLAS API.

```
#include "mtk_dense_matrix.h"
```
Include dependency graph for mtk_blas_adapter.h:
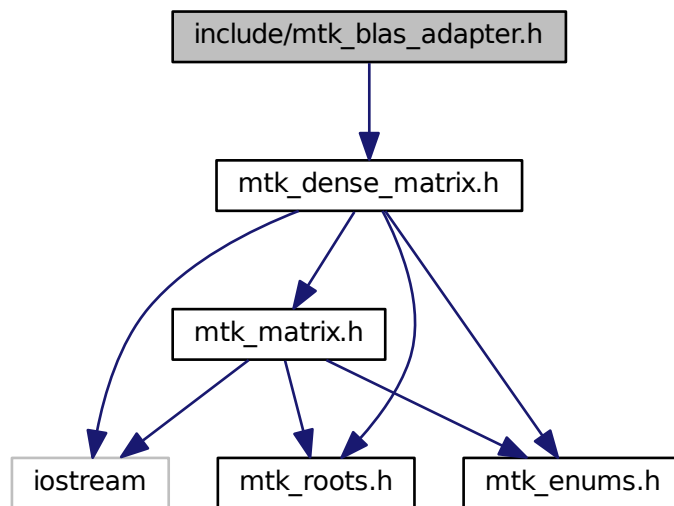
This graph shows which files directly or indirectly include this file:

Classes

- class [mtk::BLASAdapter](#)

  Adapter class for the BLAS API.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.13.1 Detailed Description

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

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

The BLAS can be installed from links given in the See Also section of this page.

See also

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

Author

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

Definition in file mtk_blas_adapter.h.

17.14 mtk_blas_adapter.h

```
00001
00024 /*
00025 Copyright (C) 2015, Computational Science Research Center, San Diego State
00026 University. All rights reserved.
00027
00028 Redistribution and use in source and binary forms, with or without modification,
00029 are permitted provided that the following conditions are met:
00030
00031 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00032 and a copy of the modified files should be reported once modifications are
00033 completed, unless these modifications are made through the project's GitHub
00034 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00035 should be developed and included in any deliverable.
00036
00037 2. Redistributions of source code must be done through direct
00038 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00039
00040 3. Redistributions in binary form must reproduce the above copyright notice,
00041 this list of conditions and the following disclaimer in the documentation and/or
00042 other materials provided with the distribution.
00043
00044 4. Usage of the binary form on proprietary applications shall require explicit
00045 prior written permission from the the copyright holders, and due credit should
00046 be given to the copyright holders.
00047
00048 5. Neither the name of the copyright holder nor the names of its contributors
00049 may be used to endorse or promote products derived from this software without
00050 specific prior written permission.
00051
00052 The copyright holders provide no reassurances that the source code provided does
00053 not infringe any patent, copyright, or any other intellectual property rights of
00054 third parties. The copyright holders disclaim any liability to any recipient for
00055 claims brought against recipient by any third party for infringement of that
00056 parties intellectual property rights.
00057
```

```
00058 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00059 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00060 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00061 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00062 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00063 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00064 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00065 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00066 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00071 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00072
00073 #include "mtk_dense_matrix.h"
00074
00075 namespace mtk {
00076
00096 class BLASAdapter {
00097  public:
00106   static Real RealNRM2(Real *in, int &in_length);
00107
00124   static void RealAXPY(Real alpha, Real *xx, Real *yy, int &in_length);
00125
00140   static Real RelNorm2Error(Real *computed, Real *known, int length);
00141
00159   static void RealDenseMV(Real &alpha,
00160                           DenseMatrix &aa,
00161                           Real *xx,
00162                           Real &beta,
00163                           Real *yy);
00164
00179   static DenseMatrix RealDenseMM(DenseMatrix &aa,
    DenseMatrix &bb);
00180
00195   static DenseMatrix RealDenseSM(Real alpha,
    DenseMatrix &aa);
00196 };
00197 }
00198 #endif  // End of: MTK_INCLUDE_BLAS_ADAPTER_H_
```

17.15   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:
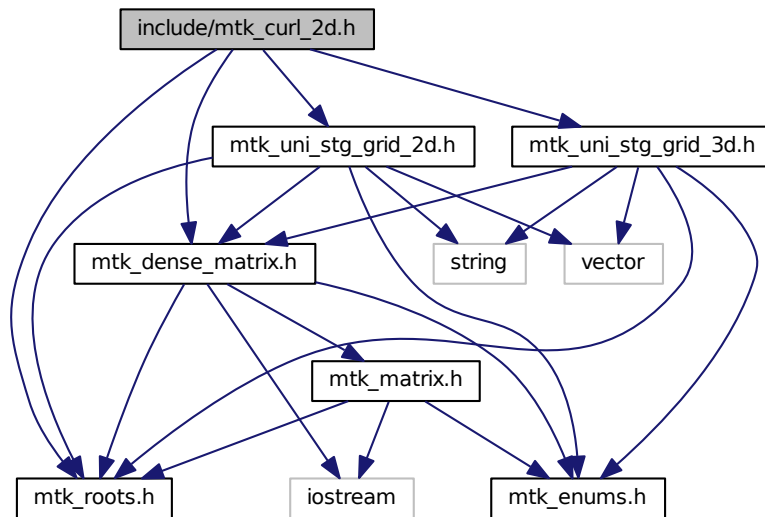
This graph shows which files directly or indirectly include this file:
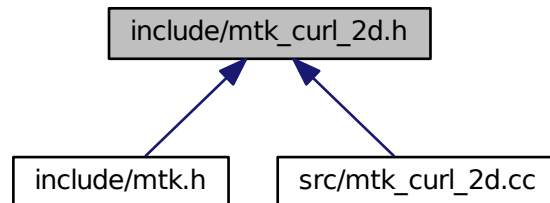
Classes

- class mtk::Curl2D

  Implements a 2D mimetic curl operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.15.1 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.

17.16 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_
```

17.17 include/mtk_dense_matrix.h File Reference

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

```
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_matrix.h"
```
Include dependency graph for mtk_dense_matrix.h:
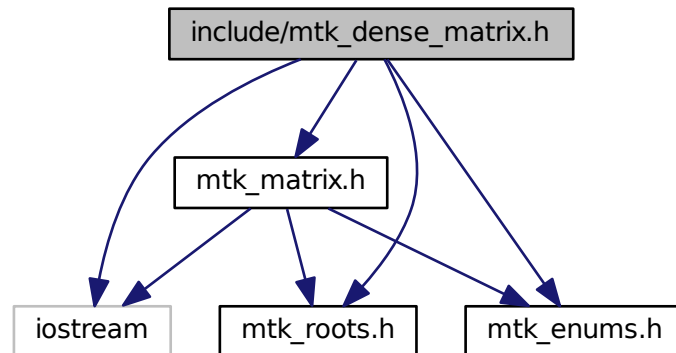
This graph shows which files directly or indirectly include this file:
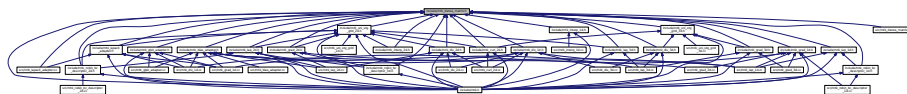
Classes

- class [mtk::DenseMatrix](#)

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

Namespaces

- [mtk](#)

    Mimetic Methods Toolkit namespace.

17.17.1 Detailed Description

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

Author

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

Note

    We prefer composition to inheritance [Reedy, 2011]. The main reason for this preference is that inheritance produces a more tightly coupled design. When a class inherits from another type be it public, protected, or private inheritance the subclass gains access to all public and protected members of the base class, whereas with composition, the class is only coupled to the public members of the other class. Furthermore, if you only hold a pointer to the other object, then your interface can use a forward declaration of the class rather than #include its full definition. This results in greater compile-time insulation and improves the time it takes to compile your code.

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

17.18 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_
```

17.19 include/mtk_div_1d.h File Reference

Includes the definition of the class Div1D.

```
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```
Include dependency graph for mtk_div_1d.h:
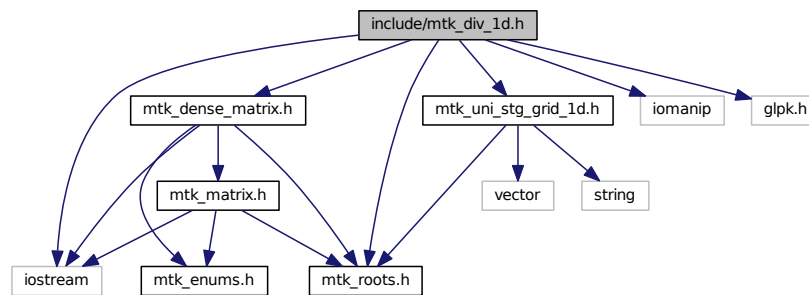
This graph shows which files directly or indirectly include this file:
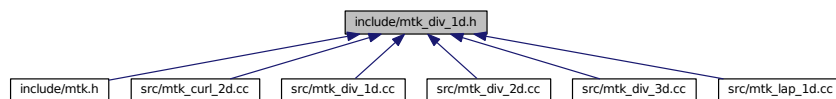
Classes

- class mtk::Div1D

  Implements a 1D mimetic divergence operator.

Namespaces

- mtk

Mimetic Methods Toolkit namespace.

17.19.1 Detailed Description

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

Author

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

Definition in file mtk_div_1d.h.

17.20 mtk_div_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_DIV_1D_H_
00058 #define MTK_INCLUDE_DIV_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
```

```
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Div1D {
00082  public:
00084    friend std::ostream& operator <<(std::ostream& stream, Div1D &in);
00085
00087    Div1D();
00088
00094    Div1D(const Div1D &div);
00095
00097    ~Div1D();
00098
00104    bool ConstructDiv1D(int order_accuracy = kDefaultOrderAccuracy,
00105                        Real mimetic_threshold = kDefaultMimeticThreshold);
00106
00112    int num_bndy_coeffs() const;
00113
00119    Real *coeffs_interior() const;
00120
00126    Real *weights_crs(void) const;
00127
00133    Real *weights_cbs(void) const;
00134
00140    DenseMatrix mim_bndy() const;
00141
00147    DenseMatrix ReturnAsDenseMatrix(const
      UniStgGrid1D &grid) const;
00148
00149  private:
00155    bool ComputeStencilInteriorGrid(void);
00156
00163    bool ComputeRationalBasisNullSpace(void);
00164
00170    bool ComputePreliminaryApproximations(void);
00171
00177    bool ComputeWeights(void);
00178
00184    bool ComputeStencilBoundaryGrid(void);
00185
00191    bool AssembleOperator(void);
00192
00193    int order_accuracy_;
00194    int dim_null_;
00195    int num_bndy_coeffs_;
00196    int divergence_length_;
00197    int minrow_;
00198    int row_;
00199
00200    DenseMatrix rat_basis_null_space_;
00201
00202    Real *coeffs_interior_;
00203    Real *prem_apps_;
00204    Real *weights_crs_;
00205    Real *weights_cbs_;
00206    Real *mim_bndy_;
00207    Real *divergence_;
00208
00209    Real mimetic_threshold_;
00210 };
00211 }
00212 #endif  // End of: MTK_INCLUDE_DIV_1D_H_
```

17.21   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:
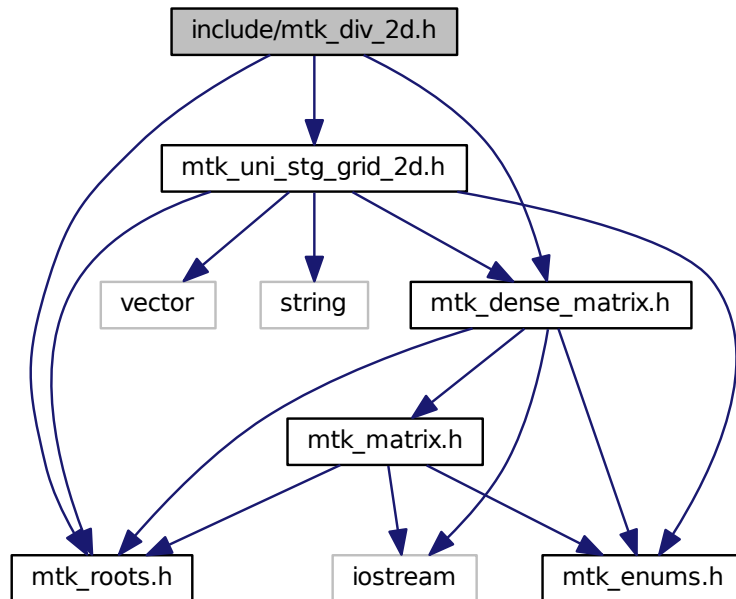
This graph shows which files directly or indirectly include this file:
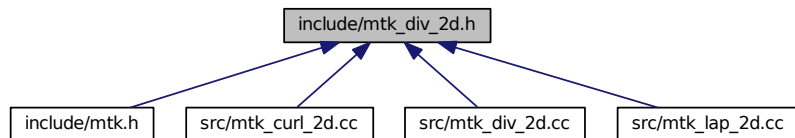
Classes

- class mtk::Div2D

  Implements a 2D mimetic divergence operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.21.1 Detailed Description

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

Author

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

Definition in file mtk_div_2d.h.

17.22 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_
```

17.23   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:
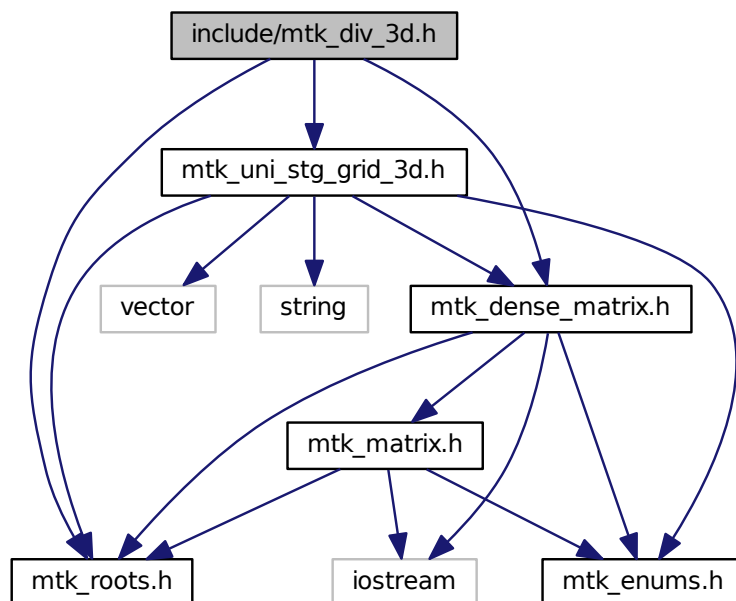
This graph shows which files directly or indirectly include this file:
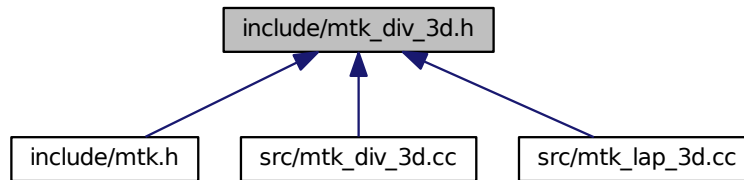
Classes

- class mtk::Div3D

  Implements a 3D mimetic divergence operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.23.1 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.

17.24 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_
```

17.25   include/mtk_enums.h File Reference

Considered enumeration types in the MTK.

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

Namespaces

- mtk
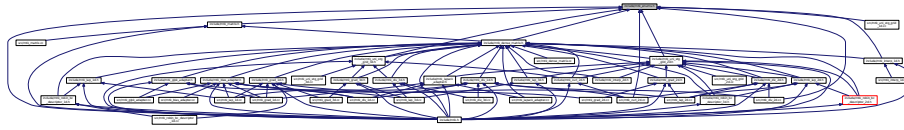
    Mimetic Methods Toolkit namespace.

Enumerations

- enum mtk::MatrixStorage { mtk::DENSE, mtk::BANDED, mtk::CRS }

    Considered matrix storage schemes to implement sparse matrices.
- enum mtk::MatrixOrdering { mtk::ROW_MAJOR, mtk::COL_MAJOR }

    Considered matrix ordering (for Fortran purposes).
- enum mtk::FieldNature { mtk::SCALAR, mtk::VECTOR }

    Nature of the field discretized in a given grid.
- enum mtk::DirInterp { mtk::SCALAR_TO_VECTOR, mtk::VECTOR_TO_SCALAR }

    Interpolation operator.

17.25.1 Detailed Description

Enumeration types are used throughout the MTK to differentiate instances of derived classes, as well as for mnemonic purposes. In this file, the enumeration types are listed alphabetically.

Author

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

Definition in file mtk_enums.h.

17.26 mtk_enums.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
```

```
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_ENUMS_H_
00059 #define MTK_INCLUDE_ENUMS_H_
00060
00061 namespace mtk {
00062
00077 enum MatrixStorage {
00078   DENSE,
00079   BANDED,
00080   CRS
00081 };
00082
00095 enum MatrixOrdering {
00096   ROW_MAJOR,
00097   COL_MAJOR
00098 };
00099
00113 enum FieldNature {
00114   SCALAR,
00115   VECTOR
00116 };
00117
00127 enum DirInterp {
00128   SCALAR_TO_VECTOR,
00129   VECTOR_TO_SCALAR
00130 };
00131 }
00132 #endif  // End of: MTK_INCLUDE_ENUMS_H_
```

17.27 include/mtk_glpk_adapter.h File Reference

Adapter class for the GLPK API.

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

Include dependency graph for mtk_glpk_adapter.h:
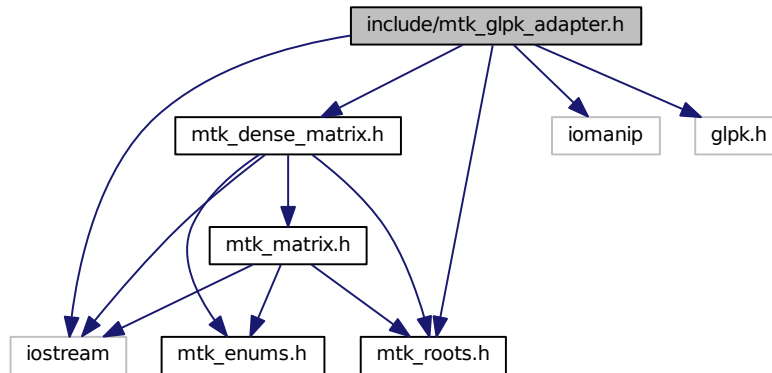
This graph shows which files directly or indirectly include this file:
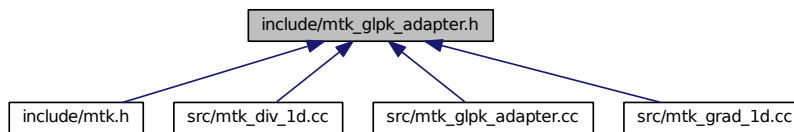
Classes

- class mtk::GLPKAdapter

    Adapter class for the GLPK API.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.27.1   Detailed Description

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

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

See also

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

Author

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

Definition in file mtk_glpk_adapter.h.

17.28   mtk_glpk_adapter.h

```
00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
00026 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00027 and a copy of the modified files should be reported once modifications are
00028 completed, unless these modifications are made through the project's GitHub
00029 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00030 should be developed and included in any deliverable.
00031
00032 2. Redistributions of source code must be done through direct
00033 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00034
00035 3. Redistributions in binary form must reproduce the above copyright notice,
00036 this list of conditions and the following disclaimer in the documentation and/or
00037 other materials provided with the distribution.
00038
00039 4. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders, and due credit should
00041 be given to the copyright holders.
00042
00043 5. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does
00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00066 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00067
00068 #include <iostream>
00069 #include <iomanip>
00070
00071 #include "glpk.h"
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_dense_matrix.h"
00075
00076 namespace mtk {
00077
00101 class GLPKAdapter {
```

```
00102  public:
00123    static mtk::Real SolveSimplexAndCompare(
      mtk::Real *A,
00124                                              int nrows,
00125                                              int ncols,
00126                                              int kk,
00127                                              mtk::Real *hh,
00128                                              mtk::Real *qq,
00129                                              int robjective,
00130                                              mtk::Real mimetic_tol,
00131                                              int copy);
00132 };
00133 }
00134 #endif  // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_
```

17.29   include/mtk_grad_1d.h File Reference

Includes the definition of the class Grad1D.

```
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```
Include dependency graph for mtk_grad_1d.h:
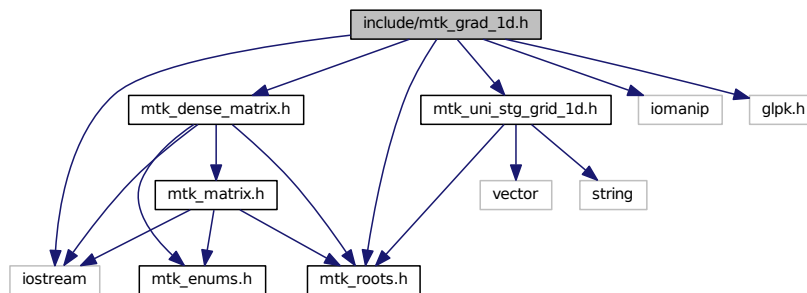
This graph shows which files directly or indirectly include this file:
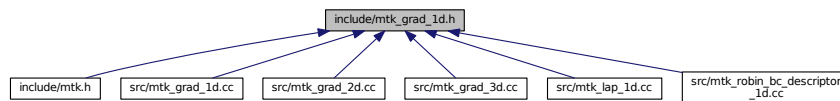
Classes

- class mtk::Grad1D

    Implements a 1D mimetic gradient operator.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.29.1 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.

17.30 mtk_grad_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_GRAD_1D_H_
00058 #define MTK_INCLUDE_GRAD_1D_H_
00059
00060 #include <iostream>
```

```
00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Grad1D {
00082  public:
00084    friend std::ostream& operator <<(std::ostream& stream, Grad1D &in);
00085
00087    Grad1D();
00088
00094    Grad1D(const Grad1D &grad);
00095
00097    ~Grad1D();
00098
00104    bool ConstructGrad1D(int order_accuracy = kDefaultOrderAccuracy,
00105                         Real mimetic_threshold = kDefaultMimeticThreshold);
00106
00112    int num_bndy_coeffs() const;
00113
00119    Real *coeffs_interior() const;
00120
00126    Real *weights_crs(void) const;
00127
00133    Real *weights_cbs(void) const;
00134
00140    DenseMatrix mim_bndy() const;
00141
00147    DenseMatrix ReturnAsDenseMatrix(Real west,
00148    Real east, int num_cells_x) const;
00154    DenseMatrix ReturnAsDenseMatrix(const
00155    UniStgGrid1D &grid) const;
00161    DenseMatrix ReturnAsDimensionlessDenseMatrix(int num_cells_x)
00162    const;
00163  private:
00169    bool ComputeStencilInteriorGrid(void);
00170
00177    bool ComputeRationalBasisNullSpace(void);
00178
00184    bool ComputePreliminaryApproximations(void);
00185
00191    bool ComputeWeights(void);
00192
00198    bool ComputeStencilBoundaryGrid(void);
00199
00205    bool AssembleOperator(void);
00206
00207    int order_accuracy_;
00208    int dim_null_;
00209    int num_bndy_approxs_;
00210    int num_bndy_coeffs_;
00211    int gradient_length_;
00212    int minrow_;
00213    int row_;
00214
00215    DenseMatrix rat_basis_null_space_;
00216
00217    Real *coeffs_interior_;
00218    Real *prem_apps_;
00219    Real *weights_crs_;
00220    Real *weights_cbs_;
00221    Real *mim_bndy_;
00222    Real *gradient_;
00223
00224    Real mimetic_threshold_;
00225 };
00226 }
00227 #endif  // End of: MTK_INCLUDE_GRAD_1D_H_
```

17.31   include/mtk_grad_2d.h File Reference

Includes the definition of the class Grad2D.

```
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_2d.h"
```
Include dependency graph for mtk_grad_2d.h:
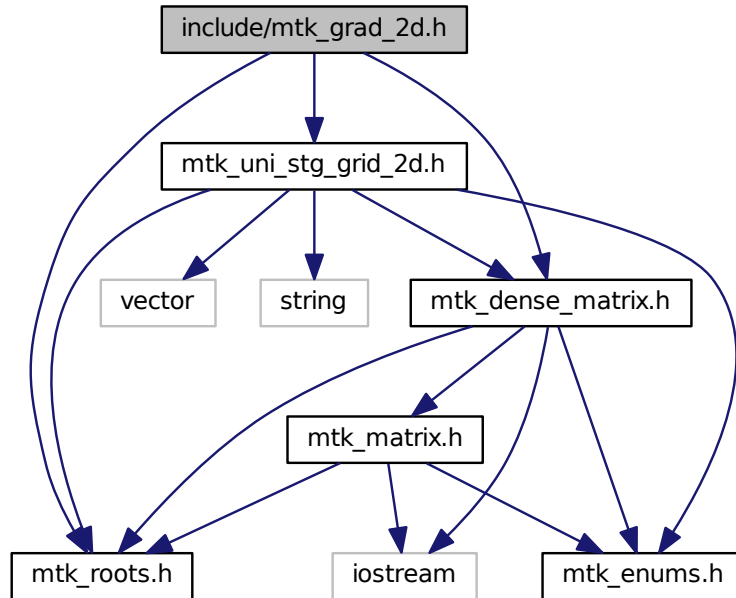
This graph shows which files directly or indirectly include this file:
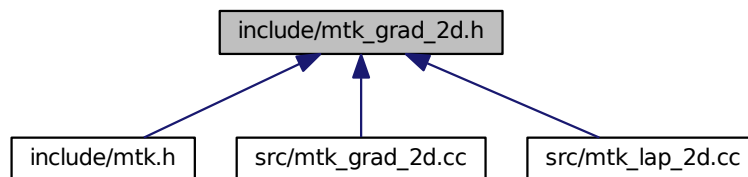
Classes

- class mtk::Grad2D

Implements a 2D mimetic gradient operator.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.31.1 Detailed Description

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

Author

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

Definition in file mtk_grad_2d.h.

17.32 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_
```

17.33 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:

```
                    ┌──────────────────────────┐
                    │  include/mtk_grad_3d.h   │
                    └──────────────────────────┘
                              │
                    ┌──────────────────────┐
                    │ mtk_uni_stg_grid_3d.h │
                    └──────────────────────┘
              ┌──────┐   ┌──────┐   ┌──────────────────┐
              │vector│   │string│   │ mtk_dense_matrix.h│
              └──────┘   └──────┘   └──────────────────┘
                              ┌──────────────┐
                              │ mtk_matrix.h │
                              └──────────────┘
          ┌─────────────┐   ┌─────────┐   ┌──────────────┐
          │ mtk_roots.h │   │ iostream│   │ mtk_enums.h  │
          └─────────────┘   └─────────┘   └──────────────┘
```

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

```
                    ┌──────────────────────────┐
                    │  include/mtk_grad_3d.h   │
                    └──────────────────────────┘
        ┌───────────────┐ ┌──────────────────┐ ┌──────────────────┐
        │ include/mtk.h │ │ src/mtk_grad_3d.cc│ │ src/mtk_lap_3d.cc│
        └───────────────┘ └──────────────────┘ └──────────────────┘
```

Classes

- class mtk::Grad3D

  Implements a 3D mimetic gradient operator.

Namespaces

- mtk

Mimetic Methods Toolkit namespace.

17.33.1 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.

17.34 mtk_grad_3d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_MTK_GRAD_3D_H_
00058 #define MTK_INCLUDE_MTK_GRAD_3D_H_
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_dense_matrix.h"
00062 #include "mtk_uni_stg_grid_3d.h"
00063
00064 namespace mtk{
```

```
00065
00076 class Grad3D {
00077  public:
00079   Grad3D();
00080
00086   Grad3D(const Grad3D &grad);
00087
00089   ~Grad3D();
00090
00096   bool ConstructGrad3D(const UniStgGrid3D &grid,
00097                        int order_accuracy = kDefaultOrderAccuracy,
00098                        Real mimetic_threshold = kDefaultMimeticThreshold);
00099
00105   DenseMatrix ReturnAsDenseMatrix() const;
00106
00107  private:
00108   DenseMatrix gradient_;
00109
00110   int order_accuracy_;
00111
00112   Real mimetic_threshold_;
00113 };
00114 }
00115 #endif  // End of: MTK_INCLUDE_MTK_GRAD_3D_H_
```

17.35   include/mtk_interp_1d.h File Reference

Includes the definition of the class Interp1D.

```
#include <iostream>
#include <iomanip>
#include "glpk.h"
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```
Include dependency graph for mtk_interp_1d.h:
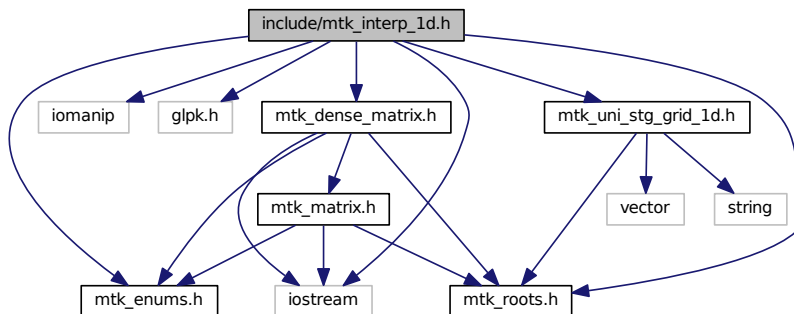
This graph shows which files directly or indirectly include this file:
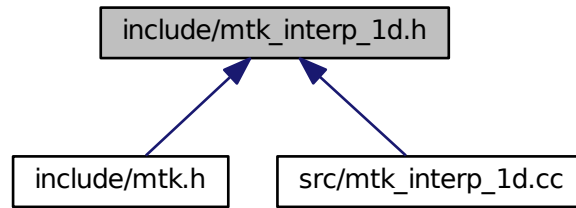
Classes

- class mtk::Interp1D

  Implements a 1D interpolation operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.35.1 Detailed Description

This class implements a 1D interpolation operator.

Author

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

Definition in file mtk_interp_1d.h.

17.36   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 =
00106 kDefaultOrderAccuracy,
00106                          mtk::DirInterp dir = SCALAR_TO_VECTOR);
00107
00113   Real *coeffs_interior() const;
00114
00120   DenseMatrix ReturnAsDenseMatrix(const
00120 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_
```

17.37 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:
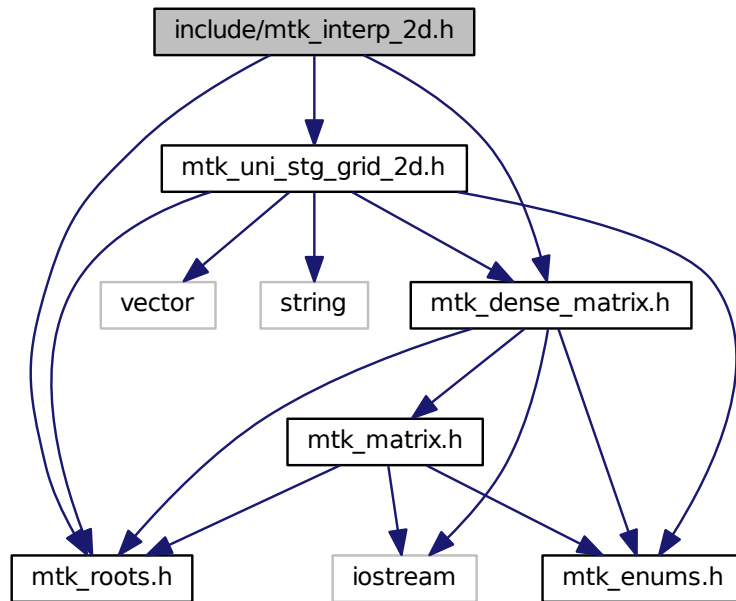
Classes

- class mtk::Interp2D

  Implements a 2D interpolation operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.37.1 Detailed Description

This class implements a 2D interpolation operator.

Author

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

Definition in file mtk_interp_2d.h.

17.38 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 =
    kDefaultMimeticThreshold);
00099
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_
```

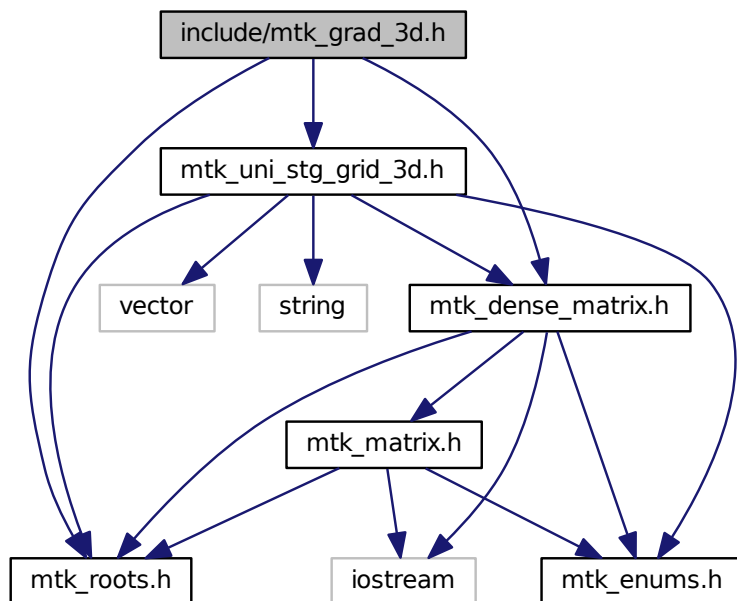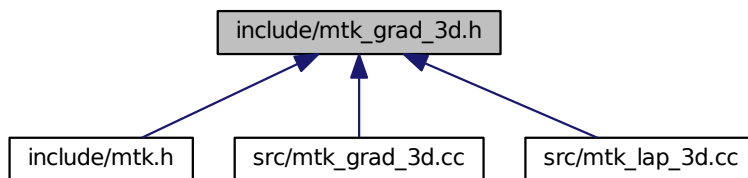## 17.39   include/mtk_lap_1d.h File Reference

Includes the definition of the class Lap1D.

```
#include "mtk_dense_matrix.h"
#include "mtk_uni_stg_grid_1d.h"
```
Include dependency graph for mtk_lap_1d.h:
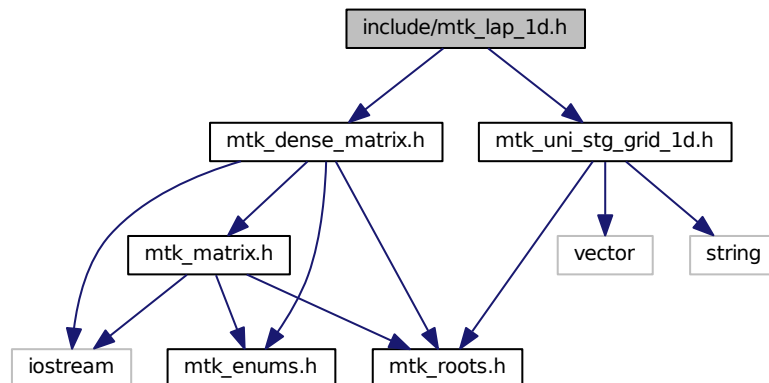
This graph shows which files directly or indirectly include this file:
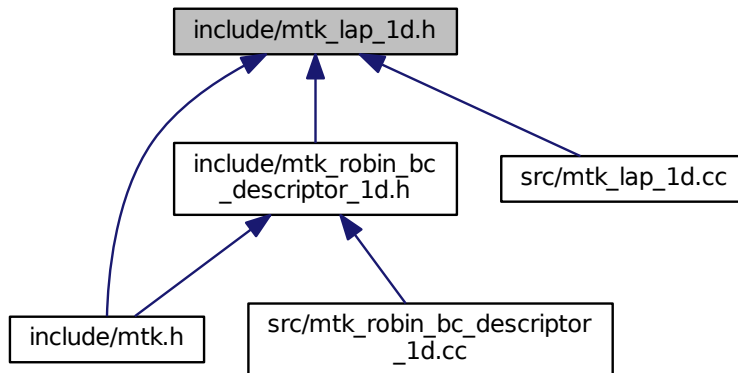
Classes

- class mtk::Lap1D

  Implements a 1D mimetic Laplacian operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.39.1 Detailed Description

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

Author

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

Definition in file mtk_lap_1d.h.

17.40   mtk_lap_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
```

```
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_LAP_1D_H_
00058 #define MTK_INCLUDE_LAP_1D_H_
00059
00060 #include "mtk_dense_matrix.h"
00061
00062 #include "mtk_uni_stg_grid_1d.h"
00063
00064 namespace mtk {
00065
00076 class Lap1D {
00077  public:
00079   friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00080
00082   Lap1D();
00083
00089   Lap1D(const Lap1D &lap);
00090
00092   ~Lap1D();
00093
00099   int order_accuracy() const;
00100
00106   Real mimetic_threshold() const;
00107
00113   Real delta() const;
00114
00120   bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00121                       Real mimetic_threshold = kDefaultMimeticThreshold);
00122
00128   DenseMatrix ReturnAsDenseMatrix(const
00      UniStgGrid1D &grid) const;
00129
00135   const mtk::Real* data(const UniStgGrid1D &grid) const;
00136
00137  private:
00138   int order_accuracy_;
00139   int laplacian_length_;
00140
00141   Real *laplacian_;
00142
00143   mutable Real delta_;
00144
00145   Real mimetic_threshold_;
```

```
00146 };
00147 }
00148 #endif  // End of: MTK_INCLUDE_LAP_1D_H_
```

17.41   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:
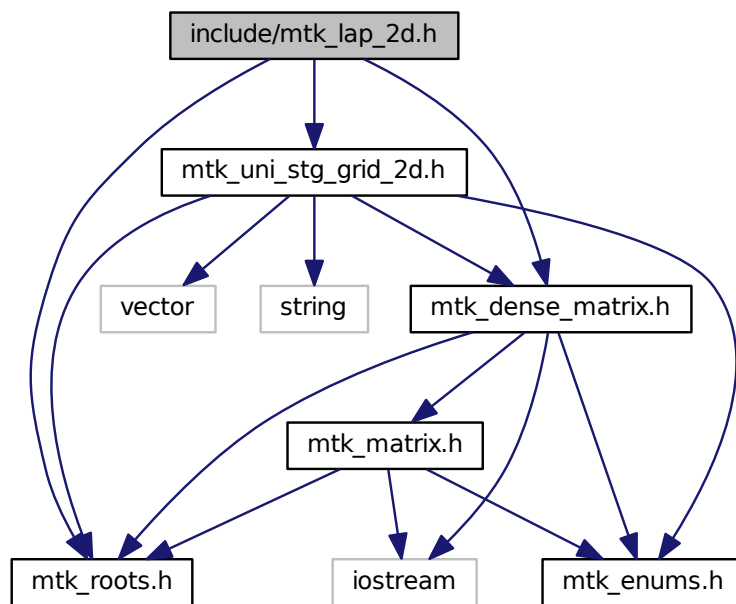
This graph shows which files directly or indirectly include this file:
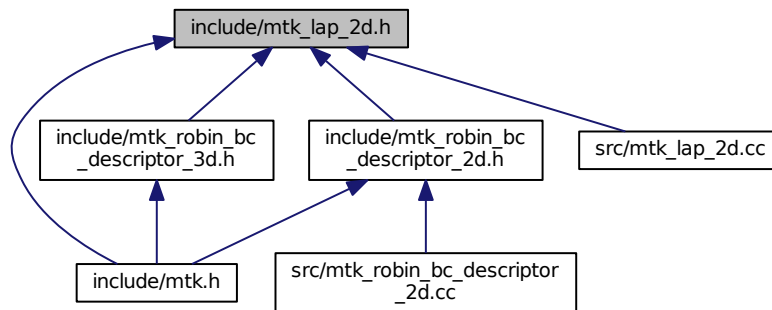
Classes

- class mtk::Lap2D

  Implements a 2D mimetic Laplacian operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.41.1 Detailed Description

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

Author

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

Definition in file mtk_lap_2d.h.

17.42 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:
00079   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_
```

17.43   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:
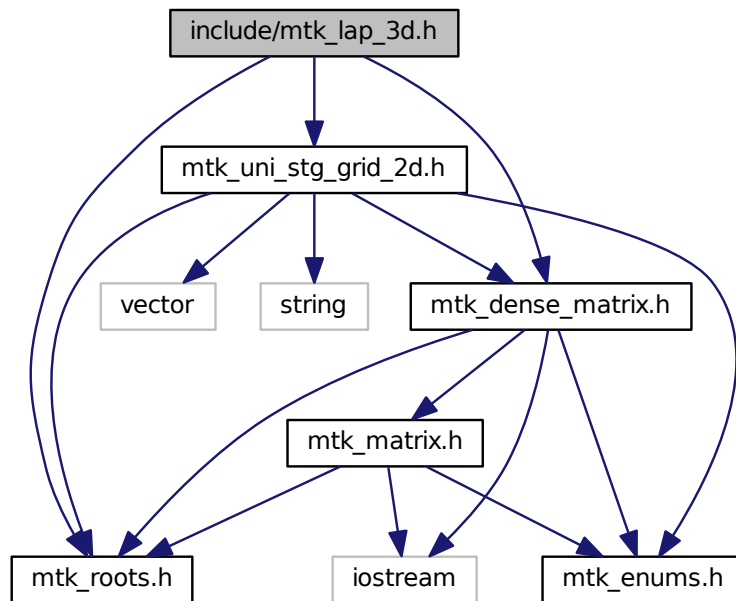
This graph shows which files directly or indirectly include this file:
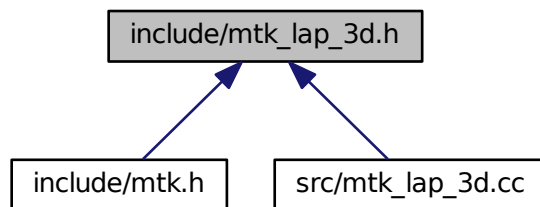
Classes

- class mtk::Lap3D

  Implements a 3D mimetic Laplacian operator.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.43.1 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.

17.44 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_
```

17.45   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:
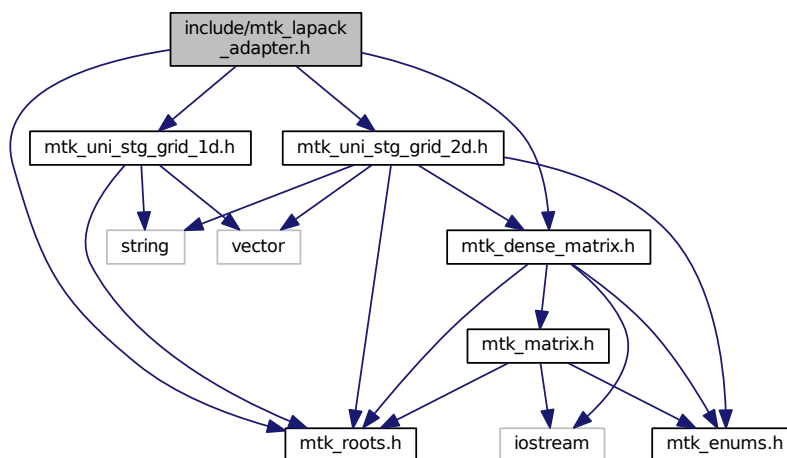
This graph shows which files directly or indirectly include this file:
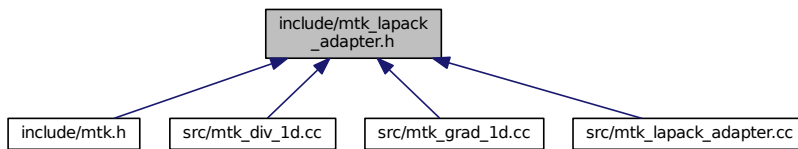
Classes

- class mtk::LAPACKAdapter

    Adapter class for the LAPACK API.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.45.1 Detailed Description

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

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

See also

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

Author

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

Definition in file mtk_lapack_adapter.h.

17.46 mtk_lapack_adapter.h

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

```
00028 completed, unless these modifications are made through the project's GitHub
00029 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00030 should be developed and included in any deliverable.
00031
00032 2. Redistributions of source code must be done through direct
00033 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00034
00035 3. Redistributions in binary form must reproduce the above copyright notice,
00036 this list of conditions and the following disclaimer in the documentation and/or
00037 other materials provided with the distribution.
00038
00039 4. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders, and due credit should
00041 be given to the copyright holders.
00042
00043 5. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does
00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00066 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00067
00068 #include "mtk_roots.h"
00069 #include "mtk_dense_matrix.h"
00070 #include "mtk_uni_stg_grid_1d.h"
00071 #include "mtk_uni_stg_grid_2d.h"
00072
00073 namespace mtk {
00074
00093 class LAPACKAdapter {
00094  public:
00105   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00106                               mtk::Real *rhs);
00107
00118   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00119                               mtk::DenseMatrix &rr);
00120
00131   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00132                               mtk::UniStgGrid1D &rhs);
00133
00134
00145   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00146                               mtk::UniStgGrid2D &rhs);
00147
00159   static int SolveRectangularDenseSystem(const
00     mtk::DenseMatrix &aa,
00160                                          mtk::Real *ob_,
00161                                          int ob_ld_);
00162
00174   static mtk::DenseMatrix QRFactorDenseMatrix(
00     DenseMatrix &matrix);
00175 };
00176 }
00177 #endif  // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_
```

17.47 include/mtk_matrix.h File Reference

Definition of the representation of a matrix in the MTK.

```
#include <iostream>
#include "mtk_roots.h"
#include "mtk_enums.h"
```
Include dependency graph for mtk_matrix.h:
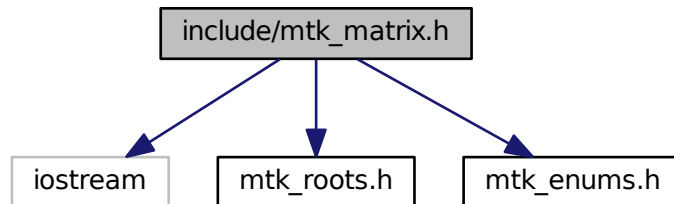
This graph shows which files directly or indirectly include this file:

Classes

- class mtk::Matrix

    Definition of the representation of a matrix in the MTK.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.47.1 Detailed Description

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

Author

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

Definition in file mtk_matrix.h.

17.48 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:
00078   Matrix();
00079
00085   Matrix(const Matrix &in);
00086
00088   ~Matrix() noexcept ;
00089
00095   MatrixStorage storage() const noexcept;
00096
00102   MatrixOrdering ordering() const noexcept;
00103
00109   int num_rows() const noexcept;
00110
00116   int num_cols() const noexcept;
00117
00123   int num_values() const noexcept;
00124
00134   int ld() const noexcept;
00135
00141   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_
```

17.49   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:

```
include/mtk_quad_1d.h
       |
   +---+---+
   |   |   |
iostream iomanip vector
```

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

```
include/mtk_quad_1d.h
         ^
         |
   include/mtk.h
```
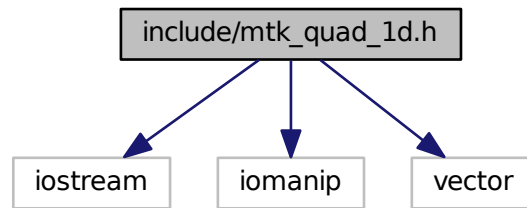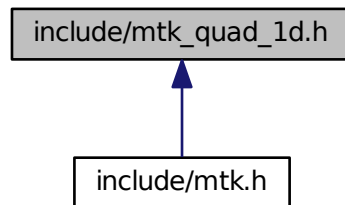
Classes

- class mtk::Quad1D

    Implements a 1D mimetic quadrature.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.49.1   Detailed Description

This class implements a 1D quadrature solver based on the mimetic discretization of the gradient operator.

See also

mtk::Grad1D

Author

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

Todo Implement this class.

Definition in file mtk_quad_1d.h.

17.50 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_
```

17.51 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:
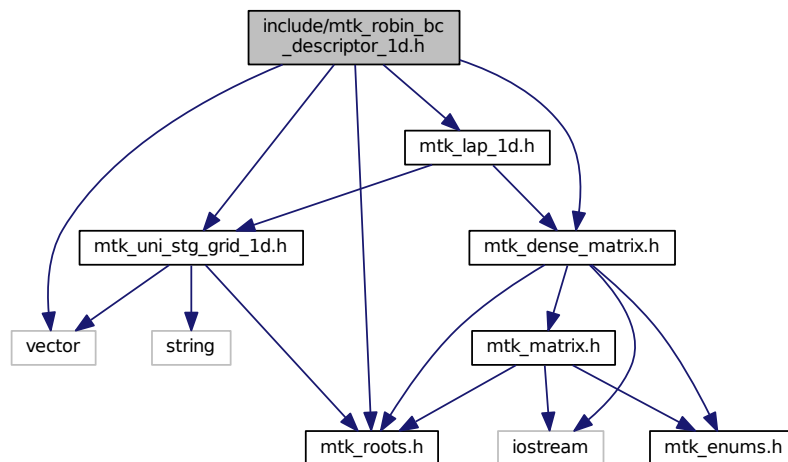
This graph shows which files directly or indirectly include this file:
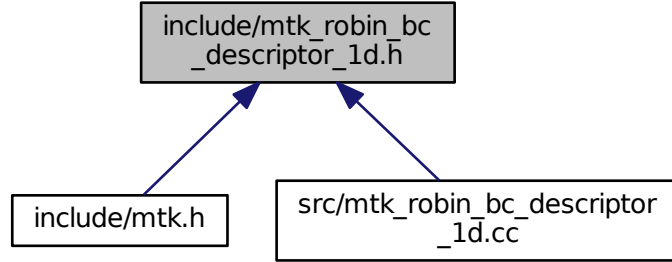
Classes

- 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.

17.51.1 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.

17.52 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_
```

17.53   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.

17.53.1 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.

17.54 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;
00262
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_
```

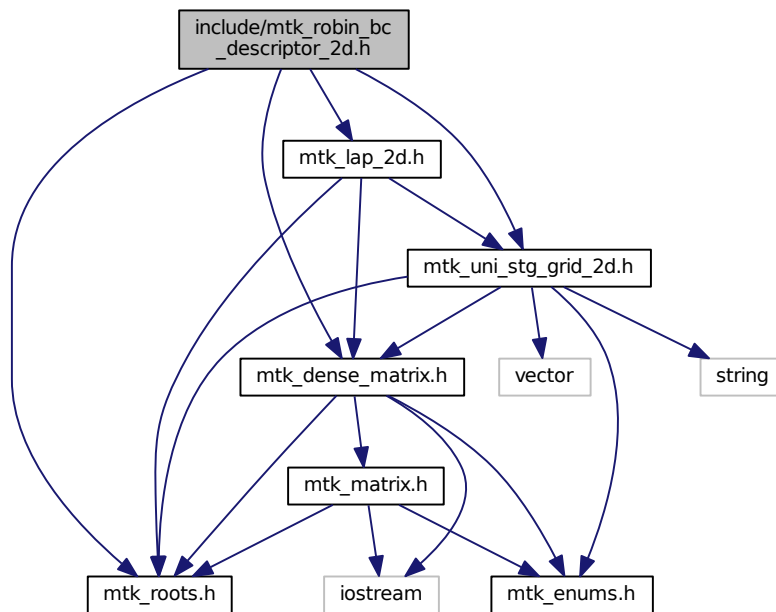## 17.55   include/mtk_robin_bc_descriptor_3d.h File Reference

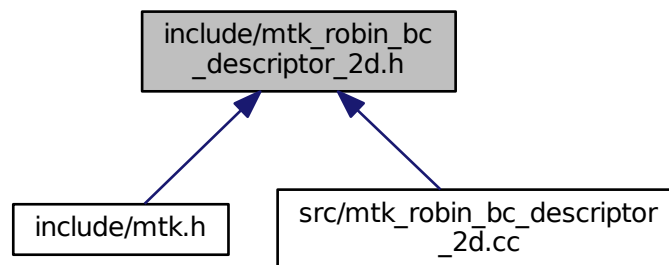Impose Robin boundary conditions on the operators and on the grids.

```
#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:
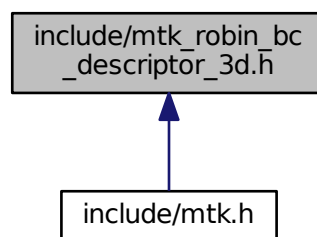
Classes

- 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.

17.55.1 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.

17.56 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
```

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_
```

17.57 include/mtk_roots.h File Reference

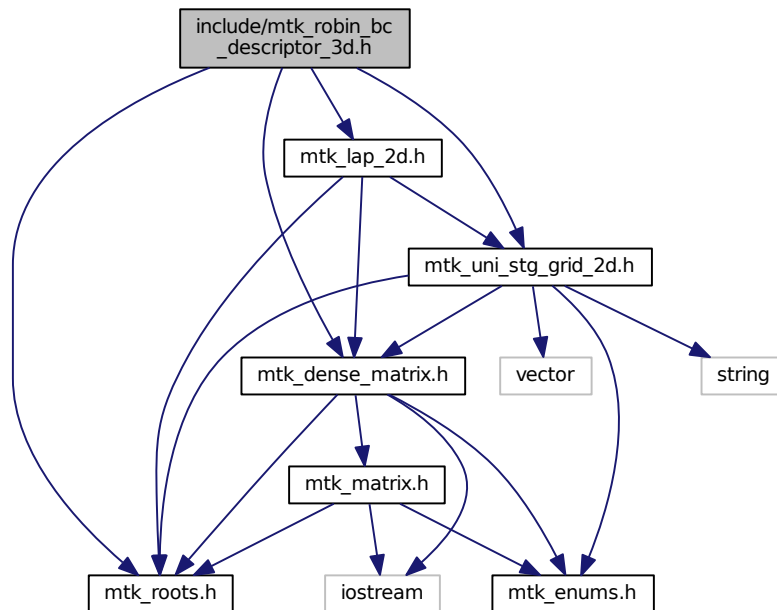Fundamental definitions to be used across all classes of the MTK.

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 int mtk::kDefaultOrderAccuracy {2}

    Default order of accuracy for mimetic operators.
- const float mtk::kDefaultMimeticThreshold {1e-6f}

    Default tolerance for higher-order mimetic operators.
- const int mtk::kCriticalOrderAccuracyDiv {8}

    At this order (and higher) we must use the CBSA to construct.
- const int mtk::kCriticalOrderAccuracyGrad {10}

    At this order (and higher) we must use the CBSA to construct.

17.57.1   Detailed Description

This file contains the fundamental definitions that classes of the MTK rely on to be implemented. Examples of these definitions are the definition of fundamental data types, and global variables affecting the construction of mimetic operators, among others.

Author

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

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

Todo   Test selective precision mechanisms.

Definition in file mtk_roots.h.

17.58   mtk_roots.h

```
00001
00017 /*
00018 Copyright (C) 2015, Computational Science Research Center, San Diego State
00019 University. All rights reserved.
00020
00021 Redistribution and use in source and binary forms, with or without modification,
00022 are permitted provided that the following conditions are met:
00023
00024 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00025 and a copy of the modified files should be reported once modifications are
00026 completed, unless these modifications are made through the project's GitHub
00027 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00028 should be developed and included in any deliverable.
00029
00030 2. Redistributions of source code must be done through direct
00031 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00032
00033 3. Redistributions in binary form must reproduce the above copyright notice,
00034 this list of conditions and the following disclaimer in the documentation and/or
00035 other materials provided with the distribution.
00036
00037 4. Usage of the binary form on proprietary applications shall require explicit
00038 prior written permission from the the copyright holders, and due credit should
00039 be given to the copyright holders.
00040
00041 5. Neither the name of the copyright holder nor the names of its contributors
00042 may be used to endorse or promote products derived from this software without
00043 specific prior written permission.
00044
00045 The copyright holders provide no reassurances that the source code provided does
00046 not infringe any patent, copyright, or any other intellectual property rights of
00047 third parties. The copyright holders disclaim any liability to any recipient for
00048 claims brought against recipient by any third party for infringement of that
00049 parties intellectual property rights.
00050
00051 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00052 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00053 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00054 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00055 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00056 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00057 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00058 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00059 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00060 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00061 */
00062
00063 #ifndef MTK_INCLUDE_ROOTS_H_
00064 #define MTK_INCLUDE_ROOTS_H_
00065
00071 namespace mtk {
00072
00080 #ifdef MTK_PRECISION_DOUBLE
00081 typedef double Real;
00082 #else
00083 typedef float Real;
00084 #endif
00085
00111 #ifdef MTK_PRECISION_DOUBLE
00112 const double kZero{0.0};
00113 const double kOne{1.0};
00114 const double kTwo{2.0};
00115 #else
00116 const float kZero{0.0f};
00117 const float kOne{1.0f};
00118 const float kTwo{2.0f};
00119 #endif
00120
00128 #ifdef MTK_PRECISION_DOUBLE
00129 const double kDefaultTolerance{1e-7};
00130 #else
00131 const float kDefaultTolerance{1e-7f};
00132 #endif
00133
00143 const int kDefaultOrderAccuracy{2};
00144
00154 #ifdef MTK_PRECISION_DOUBLE
00155 const double kDefaultMimeticThreshold{1e-6};
```

```
00156 #else
00157 const float kDefaultMimeticThreshold{1e-6f};
00158 #endif
00159
00167 const int kCriticalOrderAccuracyDiv{8};
00168
00176 const int kCriticalOrderAccuracyGrad{10};
00177 }
00178 #endif  // End of: MTK_INCLUDE_ROOTS_H_
```

17.59   include/mtk_tools.h File Reference

Tool manager class.

```
#include <ctime>
#include "mtk_roots.h"
```
Include dependency graph for mtk_tools.h:
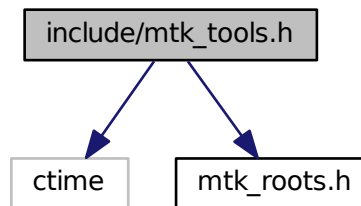
This graph shows which files directly or indirectly include this file:
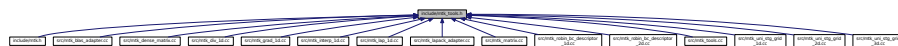
Classes

- class mtk::Tools

    Tool manager class.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.59.1   Detailed Description

Basic utilities.

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.

17.60   mtk_tools.h

```
00001
00014 /*
00015 Copyright (C) 2015, Computational Science Research Center, San Diego State
00016 University. All rights reserved.
00017
00018 Redistribution and use in source and binary forms, with or without modification,
00019 are permitted provided that the following conditions are met:
00020
00021 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00022 and a copy of the modified files should be reported once modifications are
00023 completed, unless these modifications are made through the project's GitHub
00024 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00025 should be developed and included in any deliverable.
00026
00027 2. Redistributions of source code must be done through direct
00028 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00029
00030 3. Redistributions in binary form must reproduce the above copyright notice,
00031 this list of conditions and the following disclaimer in the documentation and/or
00032 other materials provided with the distribution.
00033
00034 4. Usage of the binary form on proprietary applications shall require explicit
00035 prior written permission from the the copyright holders, and due credit should
00036 be given to the copyright holders.
00037
00038 5. Neither the name of the copyright holder nor the names of its contributors
00039 may be used to endorse or promote products derived from this software without
00040 specific prior written permission.
00041
00042 The copyright holders provide no reassurances that the source code provided does
00043 not infringe any patent, copyright, or any other intellectual property rights of
00044 third parties. The copyright holders disclaim any liability to any recipient for
00045 claims brought against recipient by any third party for infringement of that
00046 parties intellectual property rights.
00047
00048 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00049 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00050 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00051 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00052 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00053 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00054 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00055 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00056 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00057 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00058 */
00059
00060 #ifndef MTK_INCLUDE_TOOLS_H_
00061 #define MTK_INCLUDE_TOOLS_H_
00062
00063 #include <ctime>
00064
00065 #include "mtk_roots.h"
00066
00067 namespace mtk {
00068
00078 class Tools {
00079  public:
00090    static void Prevent(const bool complement,
00091                        const char *const fname,
```

```
00092                        int lineno,
00093                        const char *const fxname) noexcept;
00094
00100   static void BeginUnitTestNo(const int &nn) noexcept;
00101
00107   static void EndUnitTestNo(const int &nn) noexcept;
00108
00114   static void Assert(const bool &condition) noexcept;
00115
00116 private:
00117   static int test_number_;
00118
00119   static Real duration_;
00120
00121   static clock_t begin_time_;
00122 };
00123 }
00124 #endif  // End of: MTK_INCLUDE_TOOLS_H_
```

17.61   include/mtk_uni_stg_grid_1d.h File Reference

Definition of an 1D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_roots.h"
```
Include dependency graph for mtk_uni_stg_grid_1d.h:
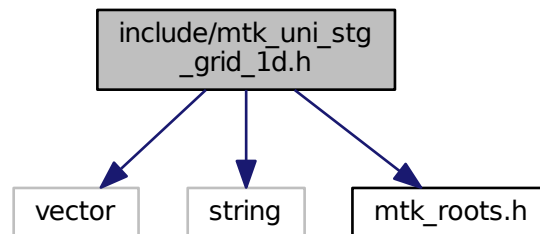
This graph shows which files directly or indirectly include this file:
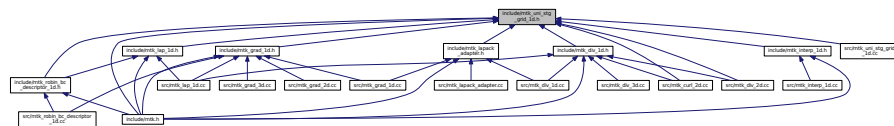
Classes

- class mtk::UniStgGrid1D

    Uniform 1D Staggered Grid.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.61.1   Detailed Description

Definition of an 1D uniform staggered grid.

Author

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

Todo Create overloaded binding routines that read data from files.

Definition in file mtk_uni_stg_grid_1d.h.

17.62   mtk_uni_stg_grid_1d.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_
```

```
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078  public:
00080   friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
00083   UniStgGrid1D();
00084
00090   UniStgGrid1D(const UniStgGrid1D &grid);
00091
00102   UniStgGrid1D(const Real &west_bndy_x,
00103                const Real &east_bndy_x,
00104                const int &num_cells_x,
00105                const mtk::FieldNature &nature = mtk::SCALAR);
00106
00108   ~UniStgGrid1D();
00109
00115   Real west_bndy_x() const;
00116
00122   Real east_bndy_x() const;
00123
00129   Real delta_x() const;
00130
00138   const Real *discrete_domain_x() const;
00139
00147   Real *discrete_field();
00148
00154   int num_cells_x() const;
00155
00161   void BindScalarField(Real (*ScalarField)(const Real &xx));
00162
00173   void BindVectorField(Real (*VectorField)(Real xx));
00174
00186   bool WriteToFile(std::string filename,
00187                    std::string space_name,
00188                    std::string field_name) const;
00189
00190  private:
00191   FieldNature nature_;
00192
00193   std::vector<Real> discrete_domain_x_;
00194   std::vector<Real> discrete_field_;
00195
00196   Real west_bndy_x_;
00197   Real east_bndy_x_;
00198   Real num_cells_x_;
00199   Real delta_x_;
00200 };
00201 }
00202 #endif  // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_
```

17.63 include/mtk_uni_stg_grid_2d.h File Reference

Definition of an 2D uniform staggered grid.

```
#include <vector>
#include <string>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_dense_matrix.h"
```

Include dependency graph for mtk_uni_stg_grid_2d.h:

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

Classes

- class [mtk::UniStgGrid2D](#)

    Uniform 2D Staggered Grid.

Namespaces

- [mtk](#)

    Mimetic Methods Toolkit namespace.

17.63.1 Detailed Description

Definition of an 2D uniform staggered grid.

Author

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

Todo Create overloaded binding routines that read data from files.

Definition in file [mtk_uni_stg_grid_2d.h](#).

17.64 mtk_uni_stg_grid_2d.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_2D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_2D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {
00069
00079 class UniStgGrid2D {
00080  public:
00082   friend std::ostream& operator <<(std::ostream& stream, UniStgGrid2D &in);
00083
00085   UniStgGrid2D();
00086
00092   UniStgGrid2D(const UniStgGrid2D &grid);
00093
00107   UniStgGrid2D(const Real &west_bndy_x,
00108                const Real &east_bndy_x,
00109                const int &num_cells_x,
00110                const Real &south_bndy_y,
00111                const Real &north_bndy_y,
00112                const int &num_cells_y,
00113                const mtk::FieldNature &nature =
00114     mtk::SCALAR);
00115
00116   ~UniStgGrid2D();
00117
```

```
00125    const Real *discrete_domain_x() const;
00126
00134    const Real *discrete_domain_y() const;
00135
00141    Real *discrete_field();
00142
00150    FieldNature nature() const;
00151
00157    Real west_bndy() const;
00158
00164    Real east_bndy() const;
00165
00171    int num_cells_x() const;
00172
00178    Real delta_x() const;
00179
00185    Real south_bndy() const;
00186
00192    Real north_bndy() const;
00193
00199    int num_cells_y() const;
00200
00206    Real delta_y() const;
00207
00213    bool Bound() const;
00214
00220    int Size() const;
00221
00227    void BindScalarField(Real (*ScalarField)(const Real &xx, const
     Real &yy));
00228
00242    void BindVectorField(Real (*VectorFieldPComponent)(const
       Real &xx,
00243                                                      const Real &yy),
00244                  Real (*VectorFieldQComponent)(const Real &xx,
00245                                                      const Real &yy));
00246
00259    bool WriteToFile(std::string filename,
00260                     std::string space_name_x,
00261                     std::string space_name_y,
00262                     std::string field_name) const;
00263
00264   private:
00276    void BindVectorFieldPComponent(
00277      Real (*VectorFieldPComponent)(const Real &xx, const Real &yy));
00278
00290    void BindVectorFieldQComponent(
00291      Real (*VectorFieldQComponent)(const Real &xx, const Real &yy));
00292
00293    std::vector<Real> discrete_domain_x_;
00294    std::vector<Real> discrete_domain_y_;
00295    std::vector<Real> discrete_field_;
00296
00297    FieldNature nature_;
00298
00299    Real west_bndy_;
00300    Real east_bndy_;
00301    int num_cells_x_;
00302    Real delta_x_;
00303
00304    Real south_bndy_;
00305    Real north_bndy_;
00306    int num_cells_y_;
00307    Real delta_y_;
00308 };
00309 }
00310 #endif  // End of: MTK_INCLUDE_UNI_STG_GRID_2D_H_
```

## 17.65   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:
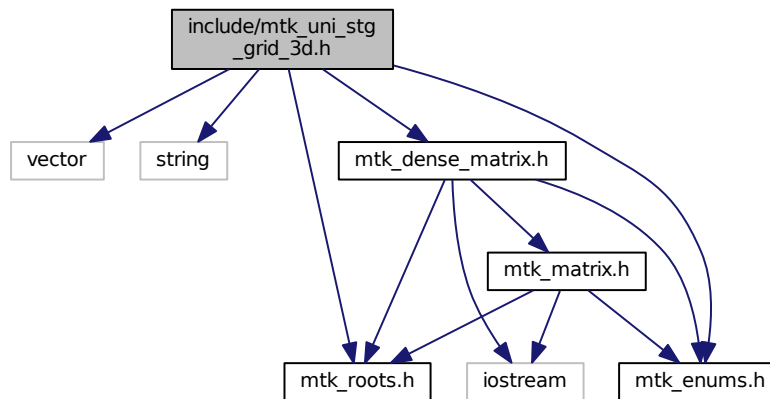
This graph shows which files directly or indirectly include this file:
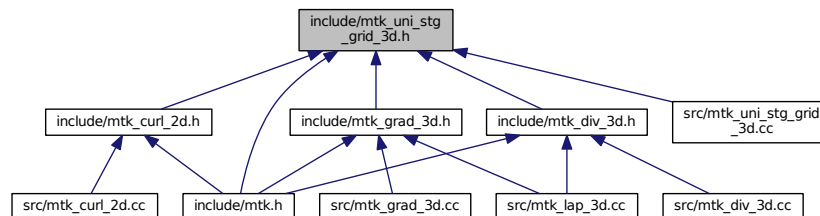
Classes

- class mtk::UniStgGrid3D

    Uniform 3D Staggered Grid.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.65.1 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.

17.66 mtk_uni_stg_grid_3d.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed, unless these modifications are made through the project's GitHub
00022 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00023 should be developed and included in any deliverable.
00024
00025 2. Redistributions of source code must be done through direct
00026 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00027
00028 3. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 4. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders, and due credit should
00034 be given to the copyright holders.
00035
00036 5. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_3D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_3D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_enums.h"
00066 #include "mtk_dense_matrix.h"
00067
00068 namespace mtk {
```

```
00069
00079 class UniStgGrid3D {
00080  public:
00082    friend std::ostream& operator <<(std::ostream& stream, UniStgGrid3D &in);
00083
00091    UniStgGrid3D operator=(const UniStgGrid3D &in);
00092
00094    UniStgGrid3D();
00095
00101    UniStgGrid3D(const UniStgGrid3D &grid);
00102
00119    UniStgGrid3D(const Real &west_bndy_x,
00120                 const Real &east_bndy_x,
00121                 const int &num_cells_x,
00122                 const Real &south_bndy_y,
00123                 const Real &north_bndy_y,
00124                 const int &num_cells_y,
00125                 const Real &bottom_bndy_z,
00126                 const Real &top_bndy_z,
00127                 const int &num_cells_z,
00128                 const mtk::FieldNature &nature =
      mtk::SCALAR);
00129
00131    ~UniStgGrid3D();
00132
00140    const Real *discrete_domain_x() const;
00141
00149    const Real *discrete_domain_y() const;
00150
00158    const Real *discrete_domain_z() const;
00159
00165    Real *discrete_field();
00166
00174    FieldNature nature() const;
00175
00181    Real west_bndy() const;
00182
00188    Real east_bndy() const;
00189
00195    int num_cells_x() const;
00196
00202    Real delta_x() const;
00203
00209    Real south_bndy() const;
00210
00216    Real north_bndy() const;
00217
00223    int num_cells_y() const;
00224
00230    Real delta_y() const;
00231
00237    Real bottom_bndy() const;
00238
00244    Real top_bndy() const;
00245
00251    int num_cells_z() const;
00252
00258    Real delta_z() const;
00259
00265    bool Bound() const;
00266
00272    int Size() const;
00273
00279    void BindScalarField(
00280      Real (*ScalarField)(const Real &xx, const Real &yy, const Real &zz));
00281
00298    void BindVectorField(Real (*VectorFieldPComponent)(const
      Real &xx,
00299                                                       const Real &yy,
00300                                                       const Real &zz),
00301                         Real (*VectorFieldQComponent)(const Real &xx,
00302                                                       const Real &yy,
00303                                                       const Real &zz),
00304                         Real (*VectorFieldRComponent)(const Real &xx,
00305                                                       const Real &yy,
00306                                                       const Real &zz));
00307
00321    bool WriteToFile(std::string filename,
00322                     std::string space_name_x,
00323                     std::string space_name_y,
00324                     std::string space_name_z,
```

```
00325                      std::string field_name) const;
00326
00327  private:
00340    void BindVectorFieldPComponent(
00341      Real (*VectorFieldPComponent)(const Real &xx,
00342                                    const Real &yy,
00343                                    const Real &zz));
00344
00357    void BindVectorFieldQComponent(
00358      Real (*VectorFieldQComponent)(const Real &xx,
00359                                    const Real &yy,
00360                                    const Real &zz));
00361
00374    void BindVectorFieldRComponent(
00375      Real (*VectorFieldRComponent)(const Real &xx,
00376                                    const Real &yy,
00377                                    const Real &zz));
00378
00379    std::vector<Real> discrete_domain_x_;
00380    std::vector<Real> discrete_domain_y_;
00381    std::vector<Real> discrete_domain_z_;
00382    std::vector<Real> discrete_field_;
00383
00384    FieldNature nature_;
00385
00386    Real west_bndy_;
00387    Real east_bndy_;
00388    int num_cells_x_;
00389    Real delta_x_;
00390
00391    Real south_bndy_;
00392    Real north_bndy_;
00393    int num_cells_y_;
00394    Real delta_y_;
00395
00396    Real bottom_bndy_;
00397    Real top_bndy_;
00398    int num_cells_z_;
00399    Real delta_z_;
00400 };
00401 }
00402 #endif  // End of: MTK_INCLUDE_UNI_STG_GRID_3D_H_
```

17.67   Makefile.inc File Reference

17.68   Makefile.inc

```
00001 # Makefile setup file for the MTK.
00002
00003 SHELL := /bin/bash
00004
00005 # Please set the following variables up:
00006
00007 #   1. Absolute path to base directory of the MTK.
00008 #   _____
00009
00010 BASE = /home/esanchez/Dropbox/MTK
00011
00012 #   2. The machine (platform) identifier and required machine precision.
00013 #   _____
00014
00015 # Options are:
00016 # - LINUX: A LINUX box installation.
00017 # - OSX: Uses OS X optimized solvers.
00018
00019 PLAT = LINUX
00020
00021 # Options are:
00022 # - SINGLE: Use 4 B floating point numbers.
00023 # - DOUBLE: Use 8 B floating point numbers.
00024
00025 PRECISION = DOUBLE
00026
00027 #   3. Optimized solvers and operations by means of ATLAS in Linux?
00028 #   _____
```

```
00029
00030 # If you have selected OSX in step 1, then you don't need to worry about this.
00031
00032 # Options are ON xor OFF:
00033
00034 ATL_OPT = OFF
00035
00036 #   4. Paths to dependencies (header files for compiling).
00037 # _____
00038
00039 # GLPK include path (soon to go):
00040
00041 GLPK_INC = $(HOME)/Libraries/glpk-4.35/include
00042
00043 # Linux: If ATLAS optimization is ON, users should only provide the path to
00044 # ATLAS:
00045
00046 ATLAS_INC = $(HOME)/Libraries/ATLAS_3.8.4-CORE/include
00047
00048 # OS X: Do nothing.
00049
00050 #   5. Paths to dependencies (archive files for (static) linking).
00051 # _____
00052
00053 # GLPK linking path (soon to go):
00054
00055 GLPK_LIB = $(HOME)/Libraries/glpk-4.35/lib/lib64/libglpk.a
00056
00057 # If optimization is OFF, then provide the paths for:
00058
00059 BLAS_LIB = $(HOME)/Libraries/BLAS-3.5.0/libblas.a
00060 LAPACK_LIB = $(HOME)/Libraries/lapack-3.5.0/liblapack.a
00061
00062 # WARNING: Vendor libraries should be used whenever they are available.
00063
00064 # However, if optimization is ON, please provide the path the ATLAS' archive:
00065
00066 ATLAS_LIB   = $(HOME)/Libraries/ATLAS_3.8.4-CORE/ATLAS_3.8.4-BUILD-Citadel/lib
00067
00068 #   6. Compiler and its flags.
00069 # _____
00070
00071 CC = g++
00072
00073 # Selective Verbose Execution for Quick Debugging. Options are defined per
00074 # concern, and per data hierarchy on each concern.
00075
00076 # 0: NO verbose at all.
00077
00078 # 1: Enable verbose down to the 7th concern: messages.
00079 # 2: Enable verbose down to the 7th concern: messages + scalar results.
00080 # 3: Enable verbose down to the 7th concern. 1.1. + array results.
00081 # 4: Enable verbose down to the 7th concern. 1.2. + matrix results.
00082
00083 # 5: Enable verbose down to the 6th concern: messages.
00084 # 6: Enable verbose down to the 6th concern: messages + scalar results.
00085 # 7: Enable verbose down to the 6th concern. 2.1. + array results.
00086 # 8: Enable verbose down to the 6th concern. 2.2. + matrix results.
00087
00088 # 9: Enable verbose down to the 5th concern: messages.
00089 # 10: Enable verbose down to the 5th concern: messages + scalar results.
00090 # 11: Enable verbose down to the 5th concern. 3.1. + array results.
00091 # 12: Enable verbose down to the 5th concern. 3.2. + matrix results.
00092
00093 # 13: Enable verbose down to the 4th concern: messages.
00094 # 14: Enable verbose down to the 4th concern: messages + scalar results.
00095 # 15: Enable verbose down to the 4th concern. 4.1. + array results.
00096 # 16: Enable verbose down to the 4th concern. 4.2. + matrix results.
00097
00098 VERBOSE_LEVEL = 16
00099
00100 # Enable preventions. In the MTK, methods first validate their required
00101 # pre-conditions in run-time. Similarly, in many points throughout the MTK
00102 # codebase, different sanity checks are performed, as well. If this symbol is
00103 # defined to be 0, the MTK will # perform no validations to enhance execution
00104 # performance. Options are:
00105 # - YES.
00106 # - NO.
00107
00108 PERFORM_PREVENTIONS = YES
00109
```

```
00110 # Flags recommended for release code:
00111
00112 CCFLAGS = -Wall -Werror -O3
00113
00114 # Flags recommended for debugging code:
00115
00116 CCFLAGS = -Wall -Werror -g
00117
00118 #   7. Archiver, its flags, and ranlib:
00119 # _____
00120
00121 ARCH      = ar
00122 ARCHFLAGS = cr
00123
00124 # If your system does not have "ranlib" then set: "RANLIB = echo":
00125
00126 RANLIB = echo
00127
00128 # But, if possible:
00129
00130 RANLIB = ranlib
00131
00132 #   8. Valgrind's memcheck options (optional):
00133 # _____
00134
00135 MEMCHECK_OPTS = -v --tool=memcheck --leak-check=full --show-leak-kinds=all \
00136   --track-origins=yes --freelist-vol=20000000
00137
00138 # Done! User, please, do not mess with the definitions from this point on.
00139
00140 # _____
00141 # _____
00142 # _____
00143
00144 #   MTK-related.
00145 # _____
00146
00147 SRC       = $(BASE)/src
00148 INCLUDE   = $(BASE)/include
00149 LIB       = $(BASE)/lib
00150 MTK_LIB   = $(LIB)/libmtk.a
00151 TESTS     = $(BASE)/tests
00152 EXAMPLES  = $(BASE)/examples
00153
00154 #   Compiling-related.
00155 # _____
00156
00157 CCFLAGS += -std=c++11 -fPIC \
00158   -DMTK_VERBOSE_LEVEL=$(VERBOSE_LEVEL) -I$(INCLUDE) -c
00159
00160 ifeq ($(PRECISION),DOUBLE)
00161   CCFLAGS += -DMTK_PRECISION_DOUBLE
00162 else
00163   CCFLAGS += -DMTK_PRECISION_SINGLE
00164 endif
00165
00166 ifeq ($(PERFORM_PREVENTIONS),YES)
00167   CCFLAGS += -DMTK_PERFORM_PREVENTIONS
00168 endif
00169
00170 # Only the GLPK is included because the other dependencies are coded in Fortran.
00171
00172 ifeq ($(ATL_OPT),ON)
00173   CCFLAGS  += -I$(GLPK_INC) $(ATLAS_INC)
00174 else
00175   CCFLAGS  += -I$(GLPK_INC)
00176 endif
00177
00178 #   Linking-related.
00179 # _____
00180
00181 NOOPT_LIBS  = $(LAPACK_LIB) $(BLAS_LIB) -lm $(GLPK_LIB) -lstdc++
00182
00183 OPT_LIBS    = -L$(ATLAS_LIB) -latlas -llapack -lblas -lm -latlas -lstdc++
00184
00185 ifeq ($(PLAT),OSX)
00186   LINKER  = g++
00187   LINKER  += -framework Accelerate $(GLPK_LIB) $(MTK_LIB)
00188 else
00189   ifeq ($(ATL_OPT),ON)
00190     LINKER  = g++
```

```
00191     LIBS = $(MTK_LIB)
00192     LIBS += $(OPT_LIBS)
00193   else
00194     LINKER  = gfortran
00195     LIBS = $(MTK_LIB)
00196     LIBS += $(NOOPT_LIBS)
00197   endif
00198 endif
00199
00200 #   Documentation-related.
00201 #   _____
00202
00203 DOCGEN     = doxygen
00204 DOCFILENAME = doc_config.dxcf
00205 DOC        = $(BASE)/doc
00206 DOCFILE    = $(BASE)/$(DOCFILENAME)
```

17.69   README.md File Reference

17.70   README.md

```
00001 # The Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, Ph.D. – esanchez at mail dot sdsu dot edu**
00004
00005 ## 1. Description
00006
00007 We define numerical methods that are based on discretizations preserving the
00008 properties of their continuum counterparts to be **mimetic**.
00009
00010 The **Mimetic Methods Toolkit (MTK)** is a C++ library for mimetic numerical
00011 methods. It is arranged as a set of classes for **mimetic quadratures**,
00012 **mimetic interpolation**, and **mimetic finite differences** methods for the
00013 numerical solution of ordinary and partial differential equations.
00014
00015 ## 2. Dependencies
00016
00017 This README assumes all of these dependencies are installed in the following
00018 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 ```
00087 ----- Library created! Check in /home/ejspeiro/Dropbox/MTK/lib
00088 ```
00089
00090 ## 4. Contact, Support, and Credits
00091
00092 The MTK is developed by researchers and adjuncts to the
00093 [Computational Science Research Center (CSRC)](http://www.csrc.sdsu.edu/)
00094 at [San Diego State University (SDSU)](http://www.sdsu.edu/).
00095
00096 Developers are members of:
00097
00098 1. Mimetic Numerical Methods Research and Development Group.
00099 2. Computational Geoscience Research and Development Group.
00100 3. Ocean Modeling Research and Development Group.
00101
00102 Currently the developers are:
00103
00104 - **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu** - @ejspeiro
00105 - Jose E. Castillo, Ph.D. - jcastillo at mail dot sdsu dot edu
00106 - Guillermo F. Miranda, Ph.D. - unigrav at hotmail dot com
00107 - Christopher P. Paolini, Ph.D. - paolini at engineering dot sdsu dot edu
00108 - Angel Boada.
00109 - Johnny Corbino.
00110 - Raul Vargas-Navarro.
00111
00112 Please reference this work as follows:
00113 ```
00114 @article{Sanchez2014308,
00115   title = "The Mimetic Methods Toolkit: An object-oriented \{API\} for Mimetic
00116 Finite Differences ",
00117   journal = "Journal of Computational and Applied Mathematics ",
00118   volume = "270",
00119   number = "",
00120   pages = "308 - 322",
00121   year = "2014",
00122   note = "Fourth International Conference on Finite Element Methods in
00123 Engineering and Sciences (FEMTEC 2013) ",
00124   issn = "0377-0427",
00125   doi = "http://dx.doi.org/10.1016/j.cam.2013.12.046",
00126   url = "http://www.sciencedirect.com/science/article/pii/S037704271300719X",
00127   author = "Eduardo J. Sanchez and Christopher P. Paolini and Jose E. Castillo",
00128   keywords = "Object-oriented development",
00129   keywords = "Partial differential equations",
00130   keywords = "Application programming interfaces",
00131   keywords = "Mimetic Finite Differences "
00132 }
00133 ```
00134
00135 Finally, please feel free to contact me with suggestions or corrections:
```

```
00136
00137 **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu** - @ejspeiro
00138
00139 Thanks and happy coding!
```

17.71 src/mtk_blas_adapter.cc File Reference

Adapter class for the BLAS API.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
```
Include dependency graph for mtk_blas_adapter.cc:

Namespaces

- mtk
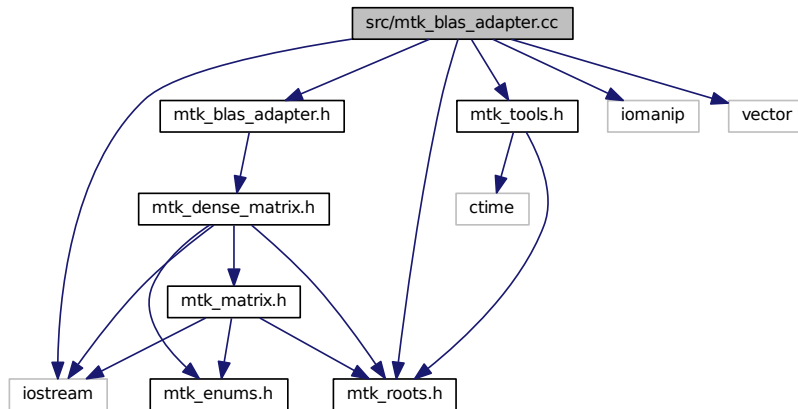
    Mimetic Methods Toolkit namespace.

Functions

- float mtk::snrm2_ (int ∗n, float ∗x, int ∗incx)
- void mtk::saxpy_ (int ∗n, float ∗sa, float ∗sx, int ∗incx, float ∗sy, int ∗incy)
- void mtk::sgemv_ (char ∗trans, int ∗m, int ∗n, float ∗alpha, float ∗a, int ∗lda, float ∗x, int ∗incx, float ∗beta, float ∗y, int ∗incy)
- void mtk::sgemm_ (char ∗transa, char ∗transb, int ∗m, int ∗n, int ∗k, double ∗alpha, double ∗a, int ∗lda, double ∗b, aamm int ∗ldb, double ∗beta, double ∗c, int ∗ldc)

17.71.1 Detailed Description

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

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

The BLAS can be installed from links given in the See Also section of this page.

See also

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

Author

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

Definition in file mtk_blas_adapter.cc.

17.72 mtk_blas_adapter.cc

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

```
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <vector>
00074
00075 #include "mtk_roots.h"
00076 #include "mtk_tools.h"
00077 #include "mtk_blas_adapter.h"
00078
00079 namespace mtk {
00080
00081 extern "C" {
00082
00083 #ifdef MTK_PRECISION_DOUBLE
00084
00097 double dnrm2_(int *n, double *x, int *incx);
00098 #else
00099
00112 float snrm2_(int *n, float *x, int *incx);
00113 #endif
00114
00115 #ifdef MTK_PRECISION_DOUBLE
00116
00135 void daxpy_(int *n, double *da, double *dx, int *incx, double *dy, int *incy);
00136 #else
00137
00156 void saxpy_(int *n, float *sa, float *sx, int *incx, float *sy, int *incy);
00157 #endif
00158
00159 #ifdef MTK_PRECISION_DOUBLE
00160
00188 void dgemv_(char *trans,
00189             int *m,
00190             int *n,
00191             double *alpha,
00192             double *a,
00193             int *lda,
00194             double *x,
00195             int *incx,
00196             double *beta,
00197             double *y,
00198             int *incy);
00199 #else
00200
00228 void sgemv_(char *trans,
00229             int *m,
00230             int *n,
00231             float *alpha,
00232             float *a,
00233             int *lda,
00234             float *x,
00235             int *incx,
00236             float *beta,
00237             float *y,
00238            int *incy);
00239 #endif
00240
00241 #ifdef MTK_PRECISION_DOUBLE
00242
00267 void dgemm_(char *transa,
00268             char* transb,
00269             int *m,
00270             int *n,
00271             int *k,
00272             double *alpha,
00273             double *a,
00274             int *lda,
00275             double *b,
00276             int *ldb,
00277             double *beta,
00278             double *c,
00279             int *ldc);
00280 }
00281 #else
00282
00307 void sgemm_(char *transa,
00308             char* transb,
00309             int *m,
```

```
00310                int *n,
00311                int *k,
00312                double *alpha,
00313                double *a,
00314                int *lda,
00315                double *b,aamm
00316                int *ldb,
00317                double *beta,
00318                double *c,
00319                int *ldc);
00320 }
00321 #endif
00322 }
00323
00324 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00325
00326   #ifdef MTK_PERFORM_PREVENTIONS
00327   mtk::Tools::Prevent(in_length <= 0, __FILE__, __LINE__, __func__);
00328   #endif
00329
00330   int incx{1};  // Increment for the elements of xx. ix >= 0.
00331
00332   #ifdef MTK_PRECISION_DOUBLE
00333   return dnrm2_(&in_length, in, &incx);
00334   #else
00335   return snrm2_(&in_length, in, &incx);
00336   #endif
00337 }
00338
00339 void mtk::BLASAdapter::RealAXPY(mtk::Real alpha,
00340                                 mtk::Real *xx,
00341                                 mtk::Real *yy,
00342                                 int &in_length) {
00343
00344   #ifdef MTK_PERFORM_PREVENTIONS
00345   mtk::Tools::Prevent(xx == nullptr, __FILE__, __LINE__, __func__);
00346   mtk::Tools::Prevent(yy == nullptr, __FILE__, __LINE__, __func__);
00347   #endif
00348
00349   int incx{1};  // Increment for the elements of xx. ix >= 0.
00350
00351   #ifdef MTK_PRECISION_DOUBLE
00352   daxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00353   #else
00354   saxpy_(&in_length, &alpha, xx, &incx, yy, &incx);
00355   #endif
00356 }
00357
00358 mtk::Real mtk::BLASAdapter::RelNorm2Error(
00359     mtk::Real *computed,
00359                                           mtk::Real *known,
00360                                           int length) {
00361
00362   #ifdef MTK_PERFORM_PREVENTIONS
00363   mtk::Tools::Prevent(computed == nullptr, __FILE__, __LINE__, __func__);
00364   mtk::Tools::Prevent(known == nullptr, __FILE__, __LINE__, __func__);
00365   #endif
00366
00367   mtk::Real norm_2_computed{mtk::BLASAdapter::RealNRM2(known, length)};
00368
00369   mtk::Real alpha{-mtk::kOne};
00370
00371   mtk::BLASAdapter::RealAXPY(alpha, known, computed, length);
00372
00373   mtk::Real norm_2_difference{mtk::BLASAdapter::RealNRM2(computed,
00373     length)};
00374
00375   return norm_2_difference/norm_2_computed;
00376 }
00377
00378 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00379                                    mtk::DenseMatrix &aa,
00380                                    mtk::Real *xx,
00381                                    mtk::Real &beta,
00382                                    mtk::Real *yy) {
00383
00384   // Make sure input matrices are row-major ordered.
00385
00386   if (aa.matrix_properties().ordering() ==
00386     mtk::COL_MAJOR) {
00387     aa.OrderRowMajor();
```

```
00388   }
00389
00390   char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00391
00392   int mm{aa.num_rows()};                    // Rows of aa.
00393   int nn{aa.num_cols()};                    // Columns of aa.
00394   int lda{(aa.matrix_properties()).ld()};   // Leading dimension.
00395   int incx{1};                              // Increment of values in x.
00396   int incy{1};                              // Increment of values in y.
00397
00398   std::swap(mm,nn);
00399   #ifdef MTK_PRECISION_DOUBLE
00400   dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00401         xx, &incx, &beta, yy, &incy);
00402   #else
00403   sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00404         xx, &incx, &beta, yy, &incy);
00405   #endif
00406   std::swap(mm,nn);
00407 }
00408
00409 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
00410   mtk::DenseMatrix &aa,
                                                    mtk::DenseMatrix &bb) {
00411
00412   #ifdef MTK_PERFORM_PREVENTIONS
00413   mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00414                       __FILE__, __LINE__, __func__);
00415   #endif
00416
00418   if (aa.matrix_properties().ordering() ==
      mtk::COL_MAJOR) {
00419     aa.OrderRowMajor();
00420   }
00421   if (bb.matrix_properties().ordering() ==
      mtk::COL_MAJOR) {
00422     bb.OrderRowMajor();
00423   }
00424
00426   char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00427   char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00428
00429   int mm{aa.num_rows()};  // Rows of aa and rows of cc.
00430   int nn{bb.num_cols()};  // Cols of bb and cols of cc.
00431   int kk{aa.num_cols()};  // Cols of aa and rows of bb.
00432
00433   int cc_num_rows{mm};  // Rows of cc.
00434   int cc_num_cols{nn};  // Columns of cc.
00435
00436   int lda{std::max(1,kk)};  // Leading dimension of the aa matrix.
00437   int ldb{std::max(1,nn)};  // Leading dimension of the bb matrix.
00438   int ldc{std::max(1,mm)};  // Leading dimension of the cc matrix.
00439
00440   mtk::Real alpha{mtk::kOne}; // First scalar coefficient.
00441   mtk::Real beta{mtk::kZero}; // Second scalar coefficient.
00442
00443   mtk::DenseMatrix cc_col_maj_ord(cc_num_rows,cc_num_cols); // Output matrix.
00444
00445   cc_col_maj_ord.SetOrdering(mtk::COL_MAJOR);
00446
00448   #ifdef MTK_PRECISION_DOUBLE
00449   dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00450         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00451   #else
00452   sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00453         bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00454   #endif
00455
00456   #if MTK_VERBOSE_LEVEL > 12
00457   std::cout << "cc_col_maj_ord =" << std::endl;
00458   std::cout << cc_col_maj_ord << std::endl;
00459   #endif
00460
00461   cc_col_maj_ord.OrderRowMajor();
00462
00463   return cc_col_maj_ord;
00464 }
00465
00466 mtk::DenseMatrix mtk::BLASAdapter::RealDenseSM(
00467   mtk::Real alpha,
                                                    mtk::DenseMatrix &aa) {
```

```
00468
00469   #ifdef MTK_PERFORM_PREVENTIONS
00470   mtk::Tools::Prevent(aa.num_rows() == 0, __FILE__, __LINE__, __func__);
00471   mtk::Tools::Prevent(aa.num_cols() == 0, __FILE__, __LINE__, __func__);
00472   #endif
00473
00475   if (aa.matrix_properties().ordering() ==
     mtk::COL_MAJOR) {
00476     aa.OrderRowMajor();
00477   }
00478
00480   char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00481   char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00482
00483   int mm{aa.num_rows()};  // Rows of aa and rows of cc.
00484   int nn{aa.num_cols()};  // Cols of bb and cols of cc.
00485   int kk{aa.num_cols()};  // Cols of aa and rows of bb.
00486
00487   int lda{std::max(1,kk)};  // Leading dimension of the aa matrix.
00488   int ldb{std::max(1,nn)};  // Leading dimension of the bb matrix.
00489   int ldc{std::max(1,mm)};  // Leading dimension of the cc matrix.
00490
00491   mtk::Real beta{alpha}; // Second scalar coefficient.
00492
00493   alpha = mtk::kZero;
00494
00495   mtk::DenseMatrix alpha_aa(aa); // Output matrix.
00496
00498   #ifdef MTK_PRECISION_DOUBLE
00499   dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00500         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00501   #else
00502   sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00503         aa.data(), &ldb, &beta, alpha_aa.data(), &ldc);
00504   #endif
00505
00506   #if MTK_VERBOSE_LEVEL > 12
00507   std::cout << "alpha_aa =" << std::endl;
00508   std::cout << alpha_aa << std::endl;
00509   #endif
00510
00511   return alpha_aa;
00512 }
```

17.73   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:
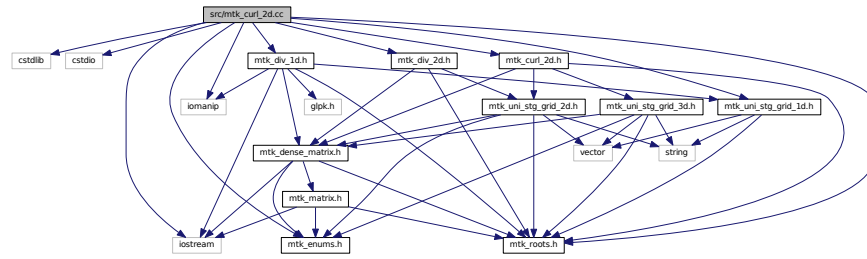
17.73.1 Detailed Description

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.

17.74 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
00073
00074   mtk::UniStgGrid3D output;
00075
00076   return output;
00077 }
00078
00079 mtk::Curl2D::Curl2D():
00080   order_accuracy_(),
00081   mimetic_threshold_() {}
00082
00083 mtk::Curl2D::Curl2D(const Curl2D &curl):
00084   order_accuracy_(curl.order_accuracy_),
00085   mimetic_threshold_(curl.mimetic_threshold_) {}
00086
00087 mtk::Curl2D::~Curl2D() {}
00088
00089 bool mtk::Curl2D::ConstructCurl2D(const
      mtk::UniStgGrid2D &grid,
00090                                     int order_accuracy,
00091                                     mtk::Real mimetic_threshold) {
00092
00093   int num_cells_x = grid.num_cells_x();
00094   int num_cells_y = grid.num_cells_y();
00095
00096   int mx = num_cells_x + 2;  // Dx vertical dimension.
00097   int nx = num_cells_x + 1;  // Dx horizontal dimension.
00098   int my = num_cells_y + 2;  // Dy vertical dimension.
00099   int ny = num_cells_y + 1;  // Dy horizontal dimension.
00100
00101   mtk::Div1D div;
00102
00103   bool info = div.ConstructDiv1D(order_accuracy, mimetic_threshold);
00104
00105   #ifdef MTK_PERFORM_PREVENTIONS
00106   if (!info) {
00107     std::cerr << "Mimetic div could not be built." << std::endl;
00108     return info;
00109   }
00110   #endif
00111
00112   auto west = grid.west_bndy();
00113   auto east = grid.east_bndy();
00114   auto south = grid.south_bndy();
00115   auto north = grid.east_bndy();
00116
00117   mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00118   mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00119
00120   mtk::DenseMatrix dx(div.ReturnAsDenseMatrix(grid_x));
00121   mtk::DenseMatrix dy(div.ReturnAsDenseMatrix(grid_y));
00122
00123   bool padded{true};
00124   bool transpose{false};
00125
00126   mtk::DenseMatrix ix(num_cells_x, padded, transpose);
00127   mtk::DenseMatrix iy(num_cells_y, padded, transpose);
00128
```

```
00129    mtk::DenseMatrix dxy(mtk::DenseMatrix::Kron(iy, dx));
00130    mtk::DenseMatrix dyx(mtk::DenseMatrix::Kron(dy, ix));
00131
00132    #if MTK_VERBOSE_LEVEL > 2
00133    std::cout << "Dx: " << mx << " by " << nx << std::endl;
00134    std::cout << "Iy : " << num_cells_y<< " by " << ny  << std::endl;
00135    std::cout << "Dy: " << my << " by " << ny << std::endl;
00136    std::cout << "Ix : " << num_cells_x<< " by " << nx  << std::endl;
00137    std::cout << "Curl 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
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 }
```

17.75 src/mtk_dense_matrix.cc File Reference

#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <typeinfo>
#include <algorithm>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_tools.h"

Include dependency graph for mtk_dense_matrix.cc:
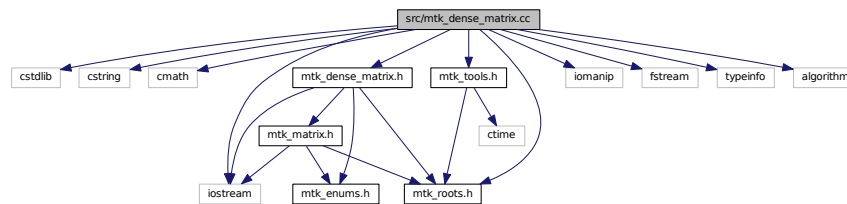
Namespaces

- mtk

Mimetic Methods Toolkit namespace.

Functions

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

17.76 mtk_dense_matrix.cc

```
00001
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed, unless these modifications are made through the project's GitHub
00023 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00024 should be developed and included in any deliverable.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 4. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders, and due credit should
00035 be given to the copyright holders.
00036
00037 5. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #include <cstdlib>
00060 #include <cstring>
00061 #include <cmath>
00062
00063 #include <iostream>
00064 #include <iomanip>
00065 #include <fstream>
00066
00067 #include <typeinfo>
00068
00069 #include <algorithm>
00070
00071 #include "mtk_roots.h"
00072 #include "mtk_dense_matrix.h"
00073 #include "mtk_tools.h"
00074
00075 namespace mtk {
00076
00077 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
```

```
00078
00079    int mm{in.matrix_properties_.num_rows()};  // Auxiliary.
00080    int nn{in.matrix_properties_.num_cols()};  // Auxiliary.
00081
00082    if (in.matrix_properties_.ordering() ==
     mtk::COL_MAJOR) {
00083      std::swap(mm, nn);
00084    }
00085    for (int ii = 0; ii < mm; ii++) {
00086      int offset{ii*nn};
00087      for (int jj = 0; jj < nn; jj++) {
00088        mtk::Real value = in.data_[offset + jj];
00089        stream << std::setw(9) << value;
00090      }
00091      stream << std::endl;
00092    }
00093    if (in.matrix_properties_.ordering() ==
     mtk::COL_MAJOR) {
00094      std::swap(mm, nn);
00095    }
00096    return stream;
00097 }
00098 }
00099
00100 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
     mtk::DenseMatrix &in) {
00101
00102    if(this == &in) {
00103      return *this;
00104    }
00105
00106    matrix_properties_.set_storage(in.
     matrix_properties_.storage());
00107
00108    matrix_properties_.set_ordering(in.
     matrix_properties_.ordering());
00109
00110    auto aux = in.matrix_properties_.num_rows();
00111    matrix_properties_.set_num_rows(aux);
00112
00113    aux = in.matrix_properties().num_cols();
00114    matrix_properties_.set_num_cols(aux);
00115
00116    aux = in.matrix_properties().num_zero();
00117    matrix_properties_.set_num_zero(aux);
00118
00119    aux = in.matrix_properties().num_null();
00120    matrix_properties_.set_num_null(aux);
00121
00122    auto num_rows = matrix_properties_.num_rows();
00123    auto num_cols = matrix_properties_.num_cols();
00124
00125    delete [] data_;
00126
00127    try {
00128      data_ = new mtk::Real[num_rows*num_cols];
00129    } catch (std::bad_alloc &memory_allocation_exception) {
00130      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00131        std::endl;
00132      std::cerr << memory_allocation_exception.what() << std::endl;
00133    }
00134    memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
     num_cols);
00135
00136    std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00137
00138    return *this;
00139 }
00140
00141 bool mtk::DenseMatrix::operator ==(const
     DenseMatrix &in) {
00142
00143    bool ans{true};
00144
00145    auto mm = in.num_rows();
00146    auto nn = in.num_cols();
00147
00148    if (mm != matrix_properties_.num_rows() ||
00149        nn != matrix_properties_.num_cols()) {
00150      return false;
00151    }
```

```
00152
00153    for (int ii = 0; ii < mm && ans; ++ii) {
00154      for (int jj = 0; jj < nn && ans; ++jj) {
00155        ans = ans &&
00156          abs(data_[ii*nn + jj] - in.data()[ii*nn + jj]) <
     mtk::kDefaultTolerance;
00157      }
00158    }
00159    return ans;
00160 }
00161
00162 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00163
00164    matrix_properties_.set_storage(mtk::DENSE);
00165    matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00166 }
00167
00168 mtk::DenseMatrix::DenseMatrix(const
     mtk::DenseMatrix &in) {
00169
00170    matrix_properties_.set_storage(in.matrix_properties_.storage());
00171
00172    matrix_properties_.set_ordering(in.matrix_properties_.
     ordering());
00173
00174    auto aux = in.matrix_properties_.num_rows();
00175    matrix_properties_.set_num_rows(aux);
00176
00177    aux = in.matrix_properties().num_cols();
00178    matrix_properties_.set_num_cols(aux);
00179
00180    aux = in.matrix_properties().num_zero();
00181    matrix_properties_.set_num_zero(aux);
00182
00183    aux = in.matrix_properties().num_null();
00184    matrix_properties_.set_num_null(aux);
00185
00186    auto num_rows = in.matrix_properties_.num_rows();
00187    auto num_cols = in.matrix_properties_.num_cols();
00188
00189    try {
00190      data_ = new mtk::Real[num_rows*num_cols];
00191    } catch (std::bad_alloc &memory_allocation_exception) {
00192      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00193        std::endl;
00194      std::cerr << memory_allocation_exception.what() << std::endl;
00195    }
00196    memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00197
00198    std::copy(in.data_,in.data_ + num_rows*num_cols,data_);
00199 }
00200
00201 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00202
00203    #ifdef MTK_PERFORM_PREVENTIONS
00204    mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00205    mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00206    #endif
00207
00208    matrix_properties_.set_storage(mtk::DENSE);
00209    matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00210    matrix_properties_.set_num_rows(num_rows);
00211    matrix_properties_.set_num_cols(num_cols);
00212
00213    try {
00214      data_ = new mtk::Real[num_rows*num_cols];
00215    } catch (std::bad_alloc &memory_allocation_exception) {
00216      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00217        std::endl;
00218      std::cerr << memory_allocation_exception.what() << std::endl;
00219    }
00220    memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00221 }
00222
00223 mtk::DenseMatrix::DenseMatrix(const int &rank,
00224                               const bool &padded,
00225                               const bool &transpose) {
00226
00227    #ifdef MTK_PERFORM_PREVENTIONS
00228    mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00229    #endif
```

```
00230
00231   int aux{};  // Used to control the padding.
00232
00233   if (padded) {
00234     aux = 1;
00235   }
00236
00237   matrix_properties_.set_storage(mtk::DENSE);
00238   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00239   matrix_properties_.set_num_rows(aux + rank + aux);
00240   matrix_properties_.set_num_cols(rank);
00241
00242   try {
00243     data_ = new mtk::Real[matrix_properties_.num_values()];
00244   } catch (std::bad_alloc &memory_allocation_exception) {
00245     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00246       std::endl;
00247     std::cerr << memory_allocation_exception.what() << std::endl;
00248   }
00249   memset(data_,
00250         mtk::kZero,
00251         sizeof(data_[0])*(matrix_properties_.num_values()));
00252
00253   for (auto ii =0 ; ii < matrix_properties_.num_rows(); ++ii) {
00254     for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00255       data_[ii*matrix_properties_.num_cols() + jj] =
00256         (ii == jj + aux)? mtk::kOne: mtk::kZero;
00257     }
00258   }
00259   if (transpose) {
00260     Transpose();
00261   }
00262 }
00263
00264 mtk::DenseMatrix::DenseMatrix(const mtk::Real *const gen,
00265                               const int &gen_length,
00266                               const int &pro_length,
00267                               const bool &transpose) {
00268
00269   #ifdef MTK_PERFORM_PREVENTIONS
00270   mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00271   mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00272   mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00273   #endif
00274
00275   matrix_properties_.set_storage(mtk::DENSE);
00276   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00277   if (!transpose) {
00278     matrix_properties_.set_num_rows(gen_length);
00279     matrix_properties_.set_num_cols(pro_length);
00280   } else {
00281     matrix_properties_.set_num_rows(pro_length);
00282     matrix_properties_.set_num_cols(gen_length);
00283   }
00284
00285   int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00286   int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00287
00288   try {
00289     data_ = new mtk::Real[mm*nn];
00290   } catch (std::bad_alloc &memory_allocation_exception) {
00291     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00292       std::endl;
00293     std::cerr << memory_allocation_exception.what() << std::endl;
00294   }
00295   memset(data_, mtk::kZero, sizeof(data_[0])*mm*nn);
00296
00297   if (!transpose) {
00298     for (auto ii = 0; ii < mm; ii++) {
00299       for (auto jj = 0; jj < nn; jj++) {
00300         data_[ii*nn + jj] = pow(gen[ii], (double) jj);
00301       }
00302     }
00303   } else {
00304     for (auto ii = 0; ii < mm; ii++) {
00305       for (auto jj = 0; jj < nn; jj++) {
00306         data_[ii*nn + jj] = pow(gen[jj], (double) ii);
00307       }
00308     }
00309   }
00310 }
```

```
00311
00312 mtk::DenseMatrix::~DenseMatrix() {
00313
00314   delete [] data_;
00315   data_ = nullptr;
00316 }
00317
00318 mtk::Matrix mtk::DenseMatrix::matrix_properties() const
      noexcept {
00319
00320   return matrix_properties_;
00321 }
00322
00323 void mtk::DenseMatrix::SetOrdering(
      mtk::MatrixOrdering oo) noexcept {
00324
00325   #ifdef MTK_PERFORM_PREVENTIONS
00326   mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
      mtk::COL_MAJOR),
00327                       __FILE__, __LINE__, __func__);
00328   #endif
00329
00330   matrix_properties_.set_ordering(oo);
00331 }
00332
00333 int mtk::DenseMatrix::num_rows() const noexcept {
00334
00335   return matrix_properties_.num_rows();
00336 }
00337
00338 int mtk::DenseMatrix::num_cols() const noexcept {
00339
00340   return matrix_properties_.num_cols();
00341 }
00342
00343 mtk::Real* mtk::DenseMatrix::data() const noexcept {
00344
00345   return data_;
00346 }
00347
00348 mtk::Real mtk::DenseMatrix::GetValue(
00349     const int &mm,
00350     const int &nn) const noexcept {
00351
00352   #ifdef MTK_PERFORM_PREVENTIONS
00353   mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00354   mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00355   #endif
00356
00357   return data_[mm*matrix_properties_.num_cols() + nn];
00358 }
00359
00360 void  mtk::DenseMatrix::SetValue(
00361     const int &mm,
00362     const int &nn,
00363     const mtk::Real &val) noexcept {
00364
00365   #ifdef MTK_PERFORM_PREVENTIONS
00366   mtk::Tools::Prevent(mm < 0, __FILE__, __LINE__, __func__);
00367   mtk::Tools::Prevent(nn < 0, __FILE__, __LINE__, __func__);
00368   #endif
00369
00370   data_[mm*matrix_properties_.num_cols() + nn] = val;
00371 }
00372
00373 void mtk::DenseMatrix::Transpose() {
00374
00376
00377   mtk::Real *data_transposed{}; // Buffer.
00378
00379   int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00380   int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00381
00382   try {
00383     data_transposed = new mtk::Real[mm*nn];
00384   } catch (std::bad_alloc &memory_allocation_exception) {
00385     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00386       std::endl;
00387     std::cerr << memory_allocation_exception.what() << std::endl;
00388   }
00389   memset(data_transposed,
```

```
00390            mtk::kZero,
00391            sizeof(data_transposed[0])*mm*nn);
00392
00393    // Assign the values to their transposed position.
00394    for (auto ii = 0; ii < mm; ++ii) {
00395      for (auto jj = 0; jj < nn; ++jj) {
00396        data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00397      }
00398    }
00399
00400    // Swap pointers.
00401    auto tmp = data_; // Temporal holder.
00402    data_ = data_transposed;
00403    delete [] tmp;
00404    tmp = nullptr;
00405
00406    matrix_properties_.set_num_rows(nn);
00407    matrix_properties_.set_num_cols(mm);
00408 }
00409
00410 void mtk::DenseMatrix::OrderRowMajor() {
00411
00412    if (matrix_properties_.ordering() == mtk::COL_MAJOR) {
00413
00415
00416      mtk::Real *data_transposed{}; // Buffer.
00417
00418      int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00419      int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00420
00421      try {
00422        data_transposed = new mtk::Real[mm*nn];
00423      } catch (std::bad_alloc &memory_allocation_exception) {
00424        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00425          std::endl;
00426        std::cerr << memory_allocation_exception.what() << std::endl;
00427      }
00428      memset(data_transposed,
00429            mtk::kZero,
00430            sizeof(data_transposed[0])*mm*nn);
00431
00432      // Assign the values to their transposed position.
00433      std::swap(mm, nn);
00434      for (auto ii = 0; ii < mm; ++ii) {
00435        for (auto jj = 0; jj < nn; ++jj) {
00436          data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00437        }
00438      }
00439      std::swap(mm, nn);
00440
00441      // Swap pointers.
00442      auto tmp = data_; // Temporal holder.
00443      data_ = data_transposed;
00444      delete [] tmp;
00445      tmp = nullptr;
00446
00447      matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00448    }
00449 }
00450
00451 void mtk::DenseMatrix::OrderColMajor() {
00452
00453    if (matrix_properties_.ordering() == ROW_MAJOR) {
00454
00456
00457      mtk::Real *data_transposed{}; // Buffer.
00458
00459      int mm = matrix_properties_.num_rows(); // Used to construct this matrix.
00460      int nn = matrix_properties_.num_cols(); // Used to construct this matrix.
00461
00462      try {
00463        data_transposed = new mtk::Real[mm*nn];
00464      } catch (std::bad_alloc &memory_allocation_exception) {
00465        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00466          std::endl;
00467        std::cerr << memory_allocation_exception.what() << std::endl;
00468      }
00469      memset(data_transposed,
00470            mtk::kZero,
00471            sizeof(data_transposed[0])*mm*nn);
00472
```

```
00473      // Assign the values to their transposed position.
00474      for (auto ii = 0; ii < mm; ++ii) {
00475        for (auto jj = 0; jj < nn; ++jj) {
00476          data_transposed[jj*mm + ii] = data_[ii*nn + jj];
00477        }
00478      }
00479
00480      // Swap pointers.
00481      auto tmp = data_; // Temporal holder.
00482      data_ = data_transposed;
00483      delete [] tmp;
00484      tmp = nullptr;
00485
00486      matrix_properties_.set_ordering(mtk::COL_MAJOR);
00487    }
00488  }
00489
00490  mtk::DenseMatrix mtk::DenseMatrix::Kron(const
      mtk::DenseMatrix &aa,
00491                                          const mtk::DenseMatrix &bb) {
00492
00493    int row_offset{}; // Offset for rows.
00494    int col_offset{}; // Offset for rows.
00495
00496    mtk::Real aa_factor{}; // Used in computation.
00497
00498    // Auxiliary variables:
00499    auto aux1 = aa.matrix_properties_.num_rows()*bb.
      matrix_properties_.num_rows();
00500    auto aux2 = aa.matrix_properties_.num_cols()*bb.
      matrix_properties_.num_cols();
00501
00502    mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00503
00504    int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00505
00506    auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00507    auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00508    auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00509    auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00510
00511    for (auto ii = 0; ii < mm; ++ii) {
00512      row_offset = ii*pp;
00513      for (auto jj = 0; jj < nn; ++jj) {
00514        col_offset = jj*qq;
00515        aa_factor = aa.data_[ii*nn + jj];
00516        for (auto ll = 0; ll < pp; ++ll) {
00517          for (auto oo = 0; oo < qq; ++oo) {
00518            auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00519            output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00520          }
00521        }
00522      }
00523    }
00524
00525    output.matrix_properties_.set_storage(mtk::DENSE);
00526    output.matrix_properties_.set_ordering(
      mtk::ROW_MAJOR);
00527
00528    return output;
00529  }
00530
00531  bool mtk::DenseMatrix::WriteToFile(const std::string &filename) const {
00532
00533    std::ofstream output_dat_file;  // Output file.
00534
00535    output_dat_file.open(filename);
00536
00537    if (!output_dat_file.is_open()) {
00538      return false;
00539    }
00540
00541    int mm{matrix_properties_.num_rows()};
00542    int nn{matrix_properties_.num_cols()};
00543
00544    for (int ii = 0; ii < mm; ++ii) {
00545      int offset{ii*nn};
00546      for (int jj = 0; jj < nn; ++jj) {
00547        output_dat_file << ii << ' ' << jj << ' ' << data_[offset + jj] <<
00548          std::endl;
00549      }
```

```
00550   }
00551
00552   output_dat_file.close();
00553
00554   return true;
00555 }
```

17.77 src/mtk_div_1d.cc File Reference

Implements the class Div1D.

```
#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_div_1d.h"
```
Include dependency graph for mtk_div_1d.cc:
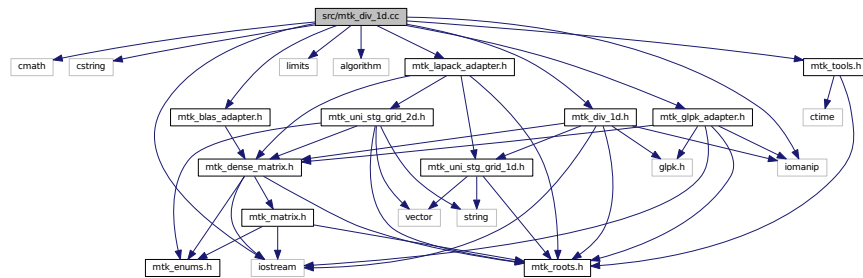
Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

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

17.77.1 Detailed Description

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

Author

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

Todo Overload ostream operator as in mtk::Lap1D.

Todo Implement creation of ■ w. mtk::BLASAdapter.

Definition in file mtk_div_1d.cc.

17.78 mtk_div_1d.cc

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066 #include <limits>
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074
```

```
00075 #include "mtk_div_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00080
00082
00083   stream << "divergence_[0] = " << std::setw(9) << in.divergence_[0] <<
00084     std::endl;
00085
00087
00088   stream << "divergence_[1:" << in.order_accuracy_ << "] = ";
00089   for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00090     stream << std::setw(9) << in.divergence_[ii] << " ";
00091   }
00092   stream << std::endl;
00093
00094   if (in.order_accuracy_ > 2) {
00095
00097
00098     stream << "divergence_[" << in.order_accuracy_ + 1 << ":" <<
00099       2*in.order_accuracy_ << "] = ";
00100     for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
    order_accuracy_; ++ii) {
00101       stream << std::setw(9) << in.divergence_[ii] << " ";
00102     }
00103     stream << std::endl;
00104
00106
00107     auto offset = (2*in.order_accuracy_ + 1);
00108     int mm{};
00109     for (auto ii = 0; ii < in.dim_null_; ++ii) {
00110       stream << "divergence_[" << offset + mm << ":" <<
00111         offset + mm + in.num_bndy_coeffs_ - 1 << "] = ";
00112       for (auto jj = 0; jj < in.num_bndy_coeffs_; ++jj) {
00113         auto value = in.divergence_[offset + mm];
00114         stream << std::setw(9) << value << " ";
00115         ++mm;
00116       }
00117       stream << std::endl;
00118     }
00119   }
00120
00121   return stream;
00122 }
00123 }
00124
00125 mtk::Div1D::Div1D():
00126   order_accuracy_(mtk::kDefaultOrderAccuracy),
00127   dim_null_(),
00128   num_bndy_coeffs_(),
00129   divergence_length_(),
00130   minrow_(),
00131   row_(),
00132   coeffs_interior_(),
00133   prem_apps_(),
00134   weights_crs_(),
00135   weights_cbs_(),
00136   mim_bndy_(),
00137   divergence_(),
00138   mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00139
00140 mtk::Div1D::Div1D(const Div1D &div):
00141   order_accuracy_(div.order_accuracy_),
00142   dim_null_(div.dim_null_),
00143   num_bndy_coeffs_(div.num_bndy_coeffs_),
00144   divergence_length_(div.divergence_length_),
00145   minrow_(div.minrow_),
00146   row_(div.row_),
00147   coeffs_interior_(div.coeffs_interior_),
00148   prem_apps_(div.prem_apps_),
00149   weights_crs_(div.weights_crs_),
00150   weights_cbs_(div.weights_cbs_),
00151   mim_bndy_(div.mim_bndy_),
00152   divergence_(div.divergence_),
00153   mimetic_threshold_(div.mimetic_threshold_) {}
00154
00155 mtk::Div1D::~Div1D() {
00156
00157   delete[] coeffs_interior_;
00158   coeffs_interior_ = nullptr;
```

```
00159
00160   delete[] prem_apps_;
00161   prem_apps_ = nullptr;
00162
00163   delete[] weights_crs_;
00164   weights_crs_ = nullptr;
00165
00166   delete[] weights_cbs_;
00167   weights_cbs_ = nullptr;
00168
00169   delete[] mim_bndy_;
00170   mim_bndy_ = nullptr;
00171
00172   delete[] divergence_;
00173   divergence_ = nullptr;
00174 }
00175
00176 bool mtk::Div1D::ConstructDiv1D(int order_accuracy,
00177                                 mtk::Real mimetic_threshold) {
00178
00179   #ifdef MTK_PERFORM_PREVENTIONS
00180   mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00181   mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00182   mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00183                       __FILE__, __LINE__, __func__);
00184
00185   if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00186     std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00187   }
00188
00189   std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00190   std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00191   #endif
00192
00193   order_accuracy_ = order_accuracy;
00194   mimetic_threshold_ = mimetic_threshold;
00195
00197
00198   bool abort_construction = ComputeStencilInteriorGrid();
00199
00200   #ifdef MTK_PERFORM_PREVENTIONS
00201   if (!abort_construction) {
00202     std::cerr << "Could NOT complete stage 1." << std::endl;
00203     std::cerr << "Exiting..." << std::endl;
00204     return false;
00205   }
00206   #endif
00207
00208   // At this point, we already have the values for the interior stencil stored
00209   // in the coeffs_interior_ array.
00210
00211   // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00212   // approximation at the boundary, thus it has no weights. For this case, the
00213   // dimension of the null-space of the Vandermonde matrices used to compute the
00214   // approximating coefficients at the boundary is 0. Ergo, we compute this
00215   // number first and then decide if we must compute anything at the boundary.
00216
00217   dim_null_ = order_accuracy_/2 - 1;
00218
00219   if (dim_null_ > 0) {
00220
00221     #ifdef MTK_PRECISION_DOUBLE
00222     num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00223     #else
00224     num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00225     #endif
00226
00228
00229     // For this we will follow recommendations given in:
00230     //
00231     // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00232     //
00233     // We will compute the QR Factorization of the transpose, as in the
00234     // following (MATLAB) pseudo-code:
00235     //
00236     // [Q,R] = qr(V'); % Full QR as defined in
00237     // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00238     //
00239     // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00240     //
00241     // However, given the nature of the Vandermonde matrices we've just
```

```
00242      // computed, they all posses the same null-space. Therefore, we impose the
00243      // convention of computing the null-space of the first Vandermonde matrix
00244      // (west boundary).
00245
00246      abort_construction = ComputeRationalBasisNullSpace();
00247
00248      #ifdef MTK_PERFORM_PREVENTIONS
00249      if (!abort_construction) {
00250        std::cerr << "Could NOT complete stage 2.1." << std::endl;
00251        std::cerr << "Exiting..." << std::endl;
00252        return false;
00253      }
00254      #endif
00255
00257
00258      abort_construction = ComputePreliminaryApproximations();
00259
00260      #ifdef MTK_PERFORM_PREVENTIONS
00261      if (!abort_construction) {
00262        std::cerr << "Could NOT complete stage 2.2." << std::endl;
00263        std::cerr << "Exiting..." << std::endl;
00264        return false;
00265      }
00266      #endif
00267
00269
00270      abort_construction = ComputeWeights();
00271
00272      #ifdef MTK_PERFORM_PREVENTIONS
00273      if (!abort_construction) {
00274        std::cerr << "Could NOT complete stage 2.3." << std::endl;
00275        std::cerr << "Exiting..." << std::endl;
00276        return false;
00277      }
00278      #endif
00279
00281
00282      abort_construction = ComputeStencilBoundaryGrid();
00283
00284      #ifdef MTK_PERFORM_PREVENTIONS
00285      if (!abort_construction) {
00286        std::cerr << "Could NOT complete stage 2.4." << std::endl;
00287        std::cerr << "Exiting..." << std::endl;
00288        return false;
00289      }
00290      #endif
00291
00292    } // End of: if (dim_null_ > 0);
00293
00295
00296    // Once we have the following three collections of data:
00297    //   (a) the coefficients for the interior,
00298    //   (b) the coefficients for the boundary (if it applies),
00299    //   (c) and the weights (if it applies),
00300    // we will store everything in the output array:
00301
00302    abort_construction = AssembleOperator();
00303
00304    #ifdef MTK_PERFORM_PREVENTIONS
00305    if (!abort_construction) {
00306      std::cerr << "Could NOT complete stage 3." << std::endl;
00307      std::cerr << "Exiting..." << std::endl;
00308      return false;
00309    }
00310    #endif
00311
00312    return true;
00313 }
00314
00315 int mtk::Div1D::num_bndy_coeffs() const {
00316
00317    return num_bndy_coeffs_;
00318 }
00319
00320 mtk::Real *mtk::Div1D::coeffs_interior() const {
00321
00322    return coeffs_interior_;
00323 }
00324
00325 mtk::Real *mtk::Div1D::weights_crs() const {
00326
```

```
00327    return weights_crs_;
00328 }
00329
00330 mtk::Real *mtk::Div1D::weights_cbs() const {
00331
00332    return weights_cbs_;
00333 }
00334
00335 mtk::DenseMatrix mtk::Div1D::mim_bndy() const {
00336
00337    mtk::DenseMatrix xx(dim_null_, 3*order_accuracy_/2);
00338
00339    auto counter = 0;
00340    for (auto ii = 0; ii < dim_null_; ++ii) {
00341      for(auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00342        xx.SetValue(ii,jj, divergence_[2*order_accuracy_ + 1 + counter]);
00343        counter++;
00344      }
00345    }
00346
00347    return xx;
00348 }
00349
00350 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(
00351    const UniStgGrid1D &grid) const {
00352
00353    int nn{grid.num_cells_x()}; // Number of cells on the grid.
00354
00355    #ifdef MTK_PERFORM_PREVENTIONS
00356    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00357    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00358    #endif
00359
00360    mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00361
00362    int dd_num_rows = nn + 2;
00363    int dd_num_cols = nn + 1;
00364    int elements_per_row = num_bndy_coeffs_;
00365    int num_extra_rows = dim_null_;
00366
00367    // Output matrix featuring sizes for divergence operators.
00368    mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00369
00370
00371
00372    auto ee_index = 0;
00373    for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00374      auto cc = 0;
00375      for(auto jj = 0 ; jj < dd_num_rows; jj++) {
00376        if( cc >= elements_per_row) {
00377          out.SetValue(ii, jj, mtk::kZero);
00378        } else {
00379          out.SetValue(ii,jj, mim_bndy_[ee_index++]*inv_delta_x);
00380          cc++;
00381        }
00382      }
00383    }
00384
00385
00386
00387    for (auto ii = num_extra_rows + 1;
00388         ii < dd_num_rows - num_extra_rows - 1; ii++) {
00389      auto jj = ii - num_extra_rows - 1;
00390      for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00391        out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00392      }
00393    }
00394
00395
00396    ee_index = 0;
00397    for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--)
00398 {
00399      auto cc = 0;
00400      for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00401        if( cc >= elements_per_row) {
00402          out.SetValue(ii,jj,0.0);
00403        } else {
00404          out.SetValue(ii,jj,-mim_bndy_[ee_index++]*inv_delta_x);
00405          cc++;
00406        }
00407      }
00408    }
00409
00410
```

```
00411    return out;
00412 }
00413
00414 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00415
00417
00418    mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00419
00420    try {
00421      pp = new mtk::Real[order_accuracy_];
00422    } catch (std::bad_alloc &memory_allocation_exception) {
00423      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00424        std::endl;
00425      std::cerr << memory_allocation_exception.what() << std::endl;
00426    }
00427    memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00428
00429    #ifdef MTK_PRECISION_DOUBLE
00430    pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00431    #else
00432    pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00433    #endif
00434
00435    for (auto ii = 1; ii < order_accuracy_; ++ii) {
00436      pp[ii] = pp[ii - 1] + mtk::kOne;
00437    }
00438
00439    #if MTK_VERBOSE_LEVEL > 3
00440    std::cout << "pp =" << std::endl;
00441    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00442      std::cout << std::setw(12) << pp[ii];
00443    }
00444    std::cout << std::endl << std::endl;
00445    #endif
00446
00448
00449    bool transpose{false};
00450
00451    mtk::DenseMatrix vander_matrix(pp,
00452                                  order_accuracy_,
00453                                  order_accuracy_,
00454                                  transpose);
00455
00456    #if MTK_VERBOSE_LEVEL > 4
00457    std::cout << "vander_matrix = " << std::endl;
00458    std::cout << vander_matrix << std::endl;
00459    #endif
00460
00462
00463    try {
00464      coeffs_interior_ = new mtk::Real[order_accuracy_];
00465    } catch (std::bad_alloc &memory_allocation_exception) {
00466      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00467        std::endl;
00468      std::cerr << memory_allocation_exception.what() << std::endl;
00469    }
00470    memset(coeffs_interior_,
00471           mtk::kZero,
00472           sizeof(coeffs_interior_[0])*order_accuracy_);
00473
00474    coeffs_interior_[1] = mtk::kOne;
00475
00476    #if MTK_VERBOSE_LEVEL > 3
00477    std::cout << "oo =" << std::endl;
00478    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00479      std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00480    }
00481    std::cout << std::endl;
00482    #endif
00483
00485
00486    int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00487                                                 coeffs_interior_)};
00488
00489    #ifdef MTK_PERFORM_PREVENTIONS
00490    if (!info) {
00491      std::cout << "System solved! Interior stencil attained!" << std::endl;
00492      std::cout << std::endl;
00493    }
00494    else {
00495      std::cerr << "Something wrong solving system! info = " << info << std::endl;
```

```
00496      std::cerr << "Exiting..." << std::endl;
00497        return false;
00498    }
00499    #endif
00500
00501    #if MTK_VERBOSE_LEVEL > 3
00502    std::cout << "coeffs_interior_ =" << std::endl;
00503    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00504      std::cout << std::setw(12) << coeffs_interior_[ii];
00505    }
00506    std::cout << std::endl << std::endl;
00507    #endif
00508
00509    delete [] pp;
00510    pp = nullptr;
00511
00512    return true;
00513 }
00514
00515 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
00516
00517    mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00518
00519
00520
00521    try {
00522      gg = new mtk::Real[num_bndy_coeffs_];
00523    } catch (std::bad_alloc &memory_allocation_exception) {
00524      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00525        std::endl;
00526      std::cerr << memory_allocation_exception.what() << std::endl;
00527    }
00528    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00529
00530    #ifdef MTK_PRECISION_DOUBLE
00531    gg[0] = -1.0/2.0;
00532    #else
00533    gg[0] = -1.0f/2.0f;
00534    #endif
00535    for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00536      gg[ii] = gg[ii - 1] + mtk::kOne;
00537    }
00538
00539    #if MTK_VERBOSE_LEVEL > 3
00540    std::cout << "gg =" << std::endl;
00541    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00542      std::cout << std::setw(12) << gg[ii];
00543    }
00544    std::cout << std::endl << std::endl;
00545    #endif
00546
00548
00549    bool tran{true}; // Should I transpose the Vandermonde matrix.
00550
00551    mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00552
00553    #if MTK_VERBOSE_LEVEL > 4
00554    std::cout << "vv_west_t =" << std::endl;
00555    std::cout << vv_west_t << std::endl;
00556    #endif
00557
00559
00560    mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
      (vv_west_t));
00561    #if MTK_VERBOSE_LEVEL > 4
00562    std::cout << "QQ^T = " << std::endl;
00563    std::cout << qq_t << std::endl;
00564    #endif
00565
00566
00568
00569    int KK_num_rows_{num_bndy_coeffs_};
00570    int KK_num_cols_{dim_null_};
00571
00572    mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00573
00574    for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00575      for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00576        KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00577              qq_t.data()[ii*num_bndy_coeffs_ + jj];
00578      }
00579    }
```

```
00580
00581   #if MTK_VERBOSE_LEVEL > 2
00582   std::cout << "KK =" << std::endl;
00583   std::cout << KK << std::endl;
00584   std::cout << "KK.num_rows() = " << KK.num_rows() << std::endl;
00585   std::cout << "KK.num_cols() = " << KK.num_cols() << std::endl;
00586   std::cout << std::endl;
00587   #endif
00588
00589
00590
00591   // Scale thus requesting that the last entries of the attained basis for the
00592   // null-space, adopt the pattern we require.
00593   // Essentially we will implement the following MATLAB pseudo-code:
00594   //   scalers = KK(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00595   //   SK = KK*scalers
00596   // where SK is the scaled null-space.
00597
00598   // In this point, we almost have all the data we need correctly allocated
00599   // in memory. We will create the matrix II_, and elements we wish to scale in
00600   // the KK array. Using the concept of the leading dimension, we could just
00601   // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00602   // GET how does it work. So I will just create a matrix with the content of
00603   // this array that we need, solve for the scalers and then scale the
00604   // whole KK:
00605
00606   // We will then create memory for that sub-matrix of KK (SUBK).
00607
00608   mtk::DenseMatrix SUBK(dim_null_,dim_null_);
00609
00610   for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00611     for (auto jj = 0; jj < dim_null_; ++jj) {
00612       SUBK.data()[(ii - (num_bndy_coeffs_ - dim_null_))*dim_null_ + jj] =
00613           KK.data()[ii*dim_null_ + jj];
00614     }
00615   }
00616
00617   #if MTK_VERBOSE_LEVEL > 4
00618   std::cout << "SUBK =" << std::endl;
00619   std::cout << SUBK << std::endl;
00620   #endif
00621
00622   SUBK.Transpose();
00623
00624   #if MTK_VERBOSE_LEVEL > 4
00625   std::cout << "SUBK^T =" << std::endl;
00626   std::cout << SUBK << std::endl;
00627   #endif
00628
00629   bool padded{false};
00630   tran = false;
00631
00632   mtk::DenseMatrix II(dim_null_, padded, tran);
00633
00634   #if MTK_VERBOSE_LEVEL > 4
00635   std::cout << "II =" << std::endl;
00636   std::cout << II << std::endl;
00637   #endif
00638
00639   // Solve the system to compute the scalers.
00640   // An example of the system to solve, for k = 8, is:
00641   //
00642   // SUBK*scalers = II_ or
00643   //
00644   // |  0.386018  -0.0339244   -0.129478 |            | 1 0 0 |
00645   // | -0.119774   0.0199423    0.0558632 |*scalers = | 0 1 0 |
00646   // | 0.0155708 -0.00349546 -0.00853182 |            | 0 0 1 |
00647   //
00648   // Notice this is a nrhs = 3 system.
00649   // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00650   // will be stored in the created identity matrix.
00651   // Let us first transpose SUBK (because of LAPACK):
00652
00653   int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00654
00655   #ifdef MTK_PERFORM_PREVENTIONS
00656   if (!info) {
00657     std::cout << "System successfully solved!" <<
00658         std::endl;
00659   } else {
00660     std::cerr << "Something went wrong solving system! info = " << info <<
00661         std::endl;
```

```
00662      std::cerr << "Exiting..." << std::endl;
00663      return false;
00664    }
00665    std::cout << std::endl;
00666    #endif
00667
00668    #if MTK_VERBOSE_LEVEL > 4
00669    std::cout << "Computed scalers:" << std::endl;
00670    std::cout << II << std::endl;
00671    #endif
00672
00673    // Multiply the two matrices to attain a scaled basis for null-space.
00674
00675    rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00676
00677    #if MTK_VERBOSE_LEVEL > 4
00678    std::cout << "Rational basis for the null-space:" << std::endl;
00679    std::cout << rat_basis_null_space_ << std::endl;
00680    #endif
00681
00682    // At this point, we have a rational basis for the null-space, with the
00683    // pattern we need! :)
00684
00685    delete [] gg;
00686    gg = nullptr;
00687
00688    return true;
00689 }
00690
00691 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00692
00693
00694
00695   mtk::Real *gg{}; // Generator vector for the first approximation.
00696
00697   try {
00698     gg = new mtk::Real[num_bndy_coeffs_];
00699   } catch (std::bad_alloc &memory_allocation_exception) {
00700     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00701 std::endl;
00702     std::cerr << memory_allocation_exception.what() << std::endl;
00703   }
00704   memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00705
00706   #ifdef MTK_PRECISION_DOUBLE
00707   gg[0] = -1.0/2.0;
00708   #else
00709   gg[0] = -1.0f/2.0f;
00710   #endif
00711   for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00712     gg[ii] = gg[ii - 1] + mtk::kOne;
00713   }
00714
00715   #if MTK_VERBOSE_LEVEL > 3
00716   std::cout << "gg0 =" << std::endl;
00717   for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00718     std::cout << std::setw(12) << gg[ii];
00719   }
00720   std::cout << std::endl << std::endl;
00721   #endif
00722
00723   // Allocate 2D array to store the collection of preliminary approximations.
00724   try {
00725     prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00726   } catch (std::bad_alloc &memory_allocation_exception) {
00727     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00728       std::endl;
00729     std::cerr << memory_allocation_exception.what() << std::endl;
00730   }
00731   memset(prem_apps_,
00732          mtk::kZero,
00733          sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00734
00736   for (auto ll = 0; ll < dim_null_; ++ll) {
00737
00738
00739     // Re-check new generator vector for every iteration except for the first.
00740     #if MTK_VERBOSE_LEVEL > 3
00741     if (ll > 0) {
00742       std::cout << "gg" << ll << " =" << std::endl;
00743       for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00744         std::cout << std::setw(12) << gg[ii];
```

```
00745        }
00746        std::cout << std::endl << std::endl;
00747      }
00748      #endif
00749
00751
00752      bool transpose{false};
00753
00754      mtk::DenseMatrix AA_(gg,
00755                          num_bndy_coeffs_, order_accuracy_ + 1,
00756                          transpose);
00757
00758      #if MTK_VERBOSE_LEVEL > 4
00759      std::cout << "AA_" << ll << " =" << std::endl;
00760      std::cout << AA_ << std::endl;
00761      #endif
00762
00764
00765      mtk::Real *ob{};
00766
00767      auto ob_ld = num_bndy_coeffs_;
00768
00769      try {
00770        ob = new mtk::Real[ob_ld];
00771      } catch (std::bad_alloc &memory_allocation_exception) {
00772        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00773          std::endl;
00774        std::cerr << memory_allocation_exception.what() << std::endl;
00775      }
00776      memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00777
00778      ob[1] = mtk::kOne;
00779
00780      #if MTK_VERBOSE_LEVEL > 4
00781      std::cout << "ob = " << std::endl << std::endl;
00782      for (auto ii = 0; ii < ob_ld; ++ii) {
00783        std::cout << std::setw(12) << ob[ii] << std::endl;
00784      }
00785      std::cout << std::endl;
00786      #endif
00787
00789
00790      // However, this is an under-determined system of equations. So we can not
00791      // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00792      // our LAPACKAdapter class.
00793
00794      int info_{
00795        mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00796  ob, ob_ld)};
00797      #ifdef MTK_PERFORM_PREVENTIONS
00798      if (!info_) {
00799        std::cout << "System successfully solved!" << std::endl << std::endl;
00800      } else {
00801        std::cerr << "Error solving system! info = " << info_ << std::endl;
00802      }
00803      #endif
00804
00805      #if MTK_VERBOSE_LEVEL > 3
00806      std::cout << "ob =" << std::endl;
00807      for (auto ii = 0; ii < ob_ld; ++ii) {
00808        std::cout << std::setw(12) << ob[ii] << std::endl;
00809      }
00810      std::cout << std::endl;
00811      #endif
00812
00814
00815      // This implies a DAXPY operation. However, we must construct the arguments
00816      // for this operation.
00817
00819      // Save them into the ob_bottom array:
00820
00821      Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00822
00823      try {
00824        ob_bottom = new mtk::Real[dim_null_];
00825      } catch (std::bad_alloc &memory_allocation_exception) {
00826        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00827          std::endl;
00828        std::cerr << memory_allocation_exception.what() << std::endl;
00829      }
```

```
00830      memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00831
00832      for (auto ii = 0; ii < dim_null_; ++ii) {
00833        ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00834      }
00835
00836      #if MTK_VERBOSE_LEVEL > 3
00837      std::cout << "ob_bottom =" << std::endl;
00838      for (auto ii = 0; ii < dim_null_; ++ii) {
00839        std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00840      }
00841      std::cout << std::endl;
00842      #endif
00843
00844
00845
00846      // We must computed an scaled ob, sob, using the scaled null-space in
00847      // rat_basis_null_space_.
00848      // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00849      // or:                 ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00850      // thus:                Y =    a*A    *x         +   b*Y (DAXPY).
00851
00852      #if MTK_VERBOSE_LEVEL > 3
00853      std::cout << "Rational basis for the null-space:" << std::endl;
00854      std::cout << rat_basis_null_space_ << std::endl;
00855      #endif
00856
00857      mtk::Real alpha{-mtk::kOne};
00858      mtk::Real beta{mtk::kOne};
00859
00860      mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00861                                    ob_bottom, beta, ob);
00862
00863      #if MTK_VERBOSE_LEVEL > 3
00864      std::cout << "scaled ob:" << std::endl;
00865      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00866        std::cout << std::setw(12) << ob[ii] << std::endl;
00867      }
00868      std::cout << std::endl;
00869      #endif
00870
00871      // We save the recently scaled solution, into an array containing these.
00872      // We can NOT start building the pi matrix, simply because I want that part
00873      // to be separated since its construction depends on the algorithm we want
00874      // to implement.
00875
00876      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00877        prem_apps_[ii*dim_null_ + ll] = ob[ii];
00878      }
00879
00880      // After the first iteration, simply shift the entries of the last
00881      // generator vector used:
00882      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00883        gg[ii]--;
00884      }
00885
00886      // Garbage collection for this loop:
00887      delete[] ob;
00888      ob = nullptr;
00889
00890      delete[] ob_bottom;
00891      ob_bottom = nullptr;
00892    } // End of: for (ll = 0; ll < dim_null; ll++);
00893
00894    #if MTK_VERBOSE_LEVEL > 4
00895    std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00896    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00897      for (auto jj = 0; jj < dim_null_; ++jj) {
00898        std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00899      }
00900      std::cout << std::endl;
00901    }
00902    std::cout << std::endl;
00903    #endif
00904
00905    delete[] gg;
00906    gg = nullptr;
00907
00908    return true;
00909 }
00910
00911 bool mtk::Div1D::ComputeWeights(void) {
```

```
00912
00913    // Matrix to copmpute the weights as in the CRSA.
00914    mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00915
00917
00918    // Assemble the pi matrix using:
00919    // 1. The collection of scaled preliminary approximations.
00920    // 2. The collection of coefficients approximating at the interior.
00921    // 3. The scaled basis for the null-space.
00922
00923    // 1.1. Process array of scaled preliminary approximations.
00924
00925    // These are queued in scaled_solutions. Each one of these, will be a column
00926    // of the pi matrix:
00927    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00928      for (auto jj = 0; jj < dim_null_; ++jj) {
00929        pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00930          prem_apps_[ii*dim_null_ + jj];
00931      }
00932    }
00933
00934    // 1.2. Add columns from known stencil approximating at the interior.
00935
00936    // However, these must be padded by zeros, according to their position in the
00937    // final pi matrix:
00938    auto mm = 0;
00939    for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
00940      for (auto ii = 0; ii < order_accuracy_; ++ii) {
00941        pi.data()[(ii + mm)*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00942          coeffs_interior_[ii];
00943      }
00944      ++mm;
00945    }
00946
00947    rat_basis_null_space_.OrderColMajor();
00948
00949    #if MTK_VERBOSE_LEVEL > 4
00950    std::cout << "Rational basis for the null-space (col. major):" << std::endl;
00951    std::cout << rat_basis_null_space_ << std::endl;
00952    #endif
00953
00954    // 1.3. Add final set of columns: rational basis for null-space.
00955    for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
00956         jj < num_bndy_coeffs_ - 1;
00957         ++jj) {
00958      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00959        auto og =
00960          (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
00961        auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
00962        pi.data()[de] = rat_basis_null_space_.data()[og];
00963      }
00964    }
00965
00966    #if MTK_VERBOSE_LEVEL > 3
00967    std::cout << "coeffs_interior_ =" << std::endl;
00968    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00969      std::cout << std::setw(12) << coeffs_interior_[ii];
00970    }
00971    std::cout << std::endl << std::endl;
00972    #endif
00973
00974    #if MTK_VERBOSE_LEVEL > 4
00975    std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
00976    std::cout << pi << std::endl;
00977    #endif
00978
00980
00981    // This imposes the mimetic condition.
00982
00983    mtk::Real *hh{};  // Right-hand side to compute weights in the C{R,B}SA.
00984
00985    try {
00986      hh = new mtk::Real[num_bndy_coeffs_];
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(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
00993
00994    hh[0] = -mtk::kOne;
```

```
00995    for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
00996      auto aux_xx = mtk::kZero;
00997      for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
00998        aux_xx += coeffs_interior_[jj];
00999      }
01000      hh[ii] = -mtk::kOne*aux_xx;
01001    }
01002
01004
01005    // That is, we construct a system, to solve for the weights.
01006
01007    // Once again we face the challenge of solving with LAPACK. However, for the
01008    // CRSA, this matrix PI is over-determined, since it has more rows than
01009    // unknowns. However, according to the theory, the solution to this system is
01010    // unique. We will use dgels_.
01011
01012    try {
01013      weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01014    } catch (std::bad_alloc &memory_allocation_exception) {
01015      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01016        std::endl;
01017      std::cerr << memory_allocation_exception.what() << std::endl;
01018    }
01019    memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01020
01021    int weights_ld{pi.num_cols() + 1};
01022
01023    // Preserve hh.
01024    std::copy(hh, hh + weights_ld, weights_cbs_);
01025
01026    pi.Transpose();
01027
01028    int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
    pi,
01029                                                    weights_cbs_,
01030                                                    weights_ld)};
01031
01032    #ifdef MTK_PERFORM_PREVENTIONS
01033    if (!info) {
01034      std::cout << "System successfully solved!" << std::endl << std::endl;
01035    } else {
01036      std::cerr << "Error solving system! info = " << info << std::endl;
01037    }
01038    #endif
01039
01040    #if MTK_VERBOSE_LEVEL > 3
01041    std::cout << "hh =" << std::endl;
01042    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01043      std::cout << std::setw(11) << hh[ii] << std::endl;
01044    }
01045    std::cout << std::endl;
01046    #endif
01047
01048    // Preserve the original weights for research.
01049
01050    try {
01051      weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01052    } catch (std::bad_alloc &memory_allocation_exception) {
01053      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01054        std::endl;
01055      std::cerr << memory_allocation_exception.what() << std::endl;
01056    }
01057    memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01058
01059    std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01060
01061    #if MTK_VERBOSE_LEVEL > 3
01062    std::cout << "weights_CRSA + lambda =" << std::endl;
01063    for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01064      std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01065    }
01066    std::cout << std::endl;
01067    #endif
01068
01070
01071    if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01072
01073      int minrow_{std::numeric_limits<int>::infinity()};
01074
01075      mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_cbs_,
    order_accuracy_)};
```

```
01076        mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01077
01078
01079
01080        mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01081
01082        for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01083          for (auto jj = 0; jj < dim_null_; ++jj) {
01084            phi.data()[ii*(order_accuracy_) + jj] = prem_apps_[ii*dim_null_ + jj];
01085          }
01086        }
01087
01088        int aux{};   // Auxiliary variable.
01089        for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01090          for (auto ii = 0; ii < order_accuracy_; ++ii) {
01091            phi.data()[(ii + aux)*order_accuracy_ + jj] = coeffs_interior_[ii];
01092          }
01093          ++aux;
01094        }
01095
01096        for(auto jj=order_accuracy_ - 1; jj >=order_accuracy_ - dim_null_; jj--) {
01097          for(auto ii=0; ii<order_accuracy_ + 1; ++ii) {
01098            phi.data()[ii*order_accuracy_+jj] = mtk::kZero;
01099          }
01100        }
01101
01102        for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01103          for (auto ii = 0; ii < dim_null_; ++ii) {
01104            phi.data()[(ii + order_accuracy_ - dim_null_ + jj*order_accuracy_)] =
01105              -prem_apps_[(dim_null_ - ii - 1 + jj*dim_null_)];
01106          }
01107        }
01108
01109        for(auto ii = 0; ii < order_accuracy_/2; ++ii) {
01110          for (auto jj = dim_null_ + 2; jj < order_accuracy_; ++jj) {
01111            auto swap = phi.data()[ii*order_accuracy_+jj];
01112            phi.data()[ii*order_accuracy_ + jj] =
01113              phi.data()[(order_accuracy_-ii)*order_accuracy_+jj];
01114            phi.data()[(order_accuracy_-ii)*order_accuracy_+jj] = swap;
01115          }
01116        }
01117
01118        #if MTK_VERBOSE_LEVEL > 4
01119        std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01120        std::cout << phi << std::endl;
01121        #endif
01122
01124
01125        mtk::Real *lamed{};   // Used to build big lambda.
01126
01127        try {
01128          lamed = new mtk::Real[dim_null_];
01129        } catch (std::bad_alloc &memory_allocation_exception) {
01130          std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01131            std::endl;
01132          std::cerr << memory_allocation_exception.what() << std::endl;
01133        }
01134        memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01135
01136        for (auto ii = 0; ii < dim_null_; ++ii) {
01137          lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01138        }
01139
01140        #if MTK_VERBOSE_LEVEL > 3
01141        std::cout << "lamed =" << std::endl;
01142        for (auto ii = 0; ii < dim_null_; ++ii) {
01143          std::cout << std::setw(12) << lamed[ii] << std::endl;
01144        }
01145        std::cout << std::endl;
01146        #endif
01147
01148        for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01149          mtk::Real temp = mtk::kZero;
01150          for(auto jj = 0; jj < dim_null_; ++jj) {
01151            temp = temp +
01152              lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01153          }
01154          hh[ii] = hh[ii] - temp;
01155        }
01156
01157        #if MTK_VERBOSE_LEVEL > 3
01158        std::cout << "big_lambda =" << std::endl;
```

```
01159      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01160        std::cout << std::setw(12) << hh[ii] << std::endl;
01161      }
01162      std::cout << std::endl;
01163      #endif
01164
01165      int copy_result{};
01166
01167      mtk::Real normerr_; // Norm of the error for the solution on each row.
01168
01169
01170
01171      for(auto row_= 0; row_ < order_accuracy_ + 1; ++row_) {
01172        normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
      data(),
01173                                                           order_accuracy_ + 1,
01174                                                           order_accuracy_,
01175                                                           order_accuracy_,
01176                                                           hh,
01177                                                           weights_cbs_,
01178                                                           row_,
01179                                                           mimetic_threshold_,
01180                                                           copy_result);
01181        mtk::Real aux{normerr_/norm_};
01182
01183        #if MTK_VERBOSE_LEVEL > 2
01184        std::cout << "Relative norm: " << aux << " " << std::endl;
01185        std::cout << std::endl;
01186        #endif
01187
01188        if (aux < minnorm_) {
01189          minnorm_ = aux;
01190          minrow_= row_;
01191        }
01192      }
01193
01194      #if MTK_VERBOSE_LEVEL > 3
01195      std::cout << "weights_CBSA + lambda (after brute force search):" <<
01196        std::endl;
01197      for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01198        std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01199      }
01200      std::cout << std::endl;
01201      #endif
01202
01203
01204      // After we know which row yields the smallest relative norm that row is
01205      // chosen to be the objective function and the result of the optimizer is
01206      // chosen to be the new weights_.
01207
01208
01209      #if MTK_VERBOSE_LEVEL > 2
01210      std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01211        minrow_ + 1 << std::endl;
01212      std::cout << std::endl;
01213      #endif
01214
01215      copy_result = 1;
01216      normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
      data(),
01217                                                         order_accuracy_ + 1,
01218                                                         order_accuracy_,
01219                                                         order_accuracy_,
01220                                                         hh,
01221                                                         weights_cbs_,
01222                                                         minrow_,
01223                                                         mimetic_threshold_,
01224                                                         copy_result);
01225      mtk::Real aux_{normerr_/norm_};
01226      #if MTK_VERBOSE_LEVEL > 2
01227      std::cout << "Relative norm: " << aux_ << std::endl;
01228      std::cout << std::endl;
01229      #endif
01230
01231      delete [] lamed;
01232      lamed = nullptr;
01233    }
01234
01235    delete [] hh;
01236    hh = nullptr;
01237
01238    return true;
01239 }
```

```
01240
01241 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01242
01243   #if MTK_VERBOSE_LEVEL > 3
01244   std::cout << "weights_CBSA + lambda =" << std::endl;
01245   for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01246     std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01247   }
01248   std::cout << std::endl;
01249   #endif
01250
01252
01253   mtk::Real *lambda{}; // Collection of bottom values from weights_.
01254
01255   try {
01256     lambda = new mtk::Real[dim_null_];
01257   } catch (std::bad_alloc &memory_allocation_exception) {
01258     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01259       std::endl;
01260     std::cerr << memory_allocation_exception.what() << std::endl;
01261   }
01262   memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01263
01264   for (auto ii = 0; ii < dim_null_; ++ii) {
01265     lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01266   }
01267
01268   #if MTK_VERBOSE_LEVEL > 3
01269   std::cout << "lambda =" << std::endl;
01270   for (auto ii = 0; ii < dim_null_; ++ii) {
01271     std::cout << std::setw(12) << lambda[ii] << std::endl;
01272   }
01273   std::cout << std::endl;
01274   #endif
01275
01277
01278   mtk::Real *alpha{}; // Collection of alpha values.
01279
01280   try {
01281     alpha = new mtk::Real[dim_null_];
01282   } catch (std::bad_alloc &memory_allocation_exception) {
01283     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01284       std::endl;
01285     std::cerr << memory_allocation_exception.what() << std::endl;
01286   }
01287   memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01288
01289   for (auto ii = 0; ii < dim_null_; ++ii) {
01290     alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01291   }
01292
01293   #if MTK_VERBOSE_LEVEL > 3
01294   std::cout << "alpha =" << std::endl;
01295   for (auto ii = 0; ii < dim_null_; ++ii) {
01296     std::cout << std::setw(12) << alpha[ii] << std::endl;
01297   }
01298   std::cout << std::endl;
01299   #endif
01300
01302
01303   try {
01304     mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01305   } catch (std::bad_alloc &memory_allocation_exception) {
01306     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01307       std::endl;
01308     std::cerr << memory_allocation_exception.what() << std::endl;
01309   }
01310   memset(mim_bndy_,
01311          mtk::kZero,
01312          sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01313
01314   for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01315     for (auto jj = 0; jj < dim_null_; ++jj) {
01316       mim_bndy_[ii*dim_null_ + jj] =
01317         prem_apps_[ii*dim_null_ + jj] +
01318         alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01319     }
01320   }
01321
01322   #if MTK_VERBOSE_LEVEL > 3
01323   std::cout << "Collection of mimetic approximations:" << std::endl;
```

```
01324    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01325      for (auto jj = 0; jj < dim_null_; ++jj) {
01326        std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01327      }
01328      std::cout << std::endl;
01329    }
01330    std::cout << std::endl;
01331    #endif
01332
01333    delete[] lambda;
01334    lambda = nullptr;
01335
01336    delete[] alpha;
01337    alpha = nullptr;
01338
01339    return true;
01340  }
01341
01342  bool mtk::Div1D::AssembleOperator(void) {
01343
01344    // The output array will have this form:
01345    // 1. The first entry of the array will contain used order order_accuracy_.
01346    // 2. The second entry of the array will contain the collection of
01347    // approximating coefficients for the interior of the grid.
01348    // 3. IF order_accuracy_ > 2, then the third entry will contain a collection
01349    // of weights.
01350    // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the
01351    // collections of approximating coefficients for the west boundary of the
01352    // grid.
01353
01354    if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01355      divergence_length_ =
01356        1 + order_accuracy_ + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01357    } else {
01358      divergence_length_ = 1 + order_accuracy_;
01359    }
01360
01361    #if MTK_VERBOSE_LEVEL > 2
01362    std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01363    #endif
01364
01365    try {
01366      divergence_ = new double[divergence_length_];
01367    } catch (std::bad_alloc &memory_allocation_exception) {
01368      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01369        std::endl;
01370      std::cerr << memory_allocation_exception.what() << std::endl;
01371    }
01372    memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01373
01375
01376    divergence_[0] = order_accuracy_;
01377
01379
01380    for (auto ii = 0; ii < order_accuracy_; ++ii) {
01381    divergence_[ii + 1] = coeffs_interior_[ii];
01382    }
01383
01385
01386    if (order_accuracy_ > 2) {
01387      for (auto ii = 0; ii < order_accuracy_; ++ii) {
01388        divergence_[(1 + order_accuracy_) + ii] = weights_cbs_[ii];
01389      }
01390    }
01391
01394
01395    if (order_accuracy_ > 2) {
01396      auto offset = (2*order_accuracy_ + 1);
01397      int mm{};
01398      for (auto ii = 0; ii < dim_null_; ++ii) {
01399        for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01400          divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01401          ++mm;
01402        }
01403      }
01404    }
01405
01406    #if MTK_VERBOSE_LEVEL > 1
01407    std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01408    std::cout << std::endl;
01409    #endif
```

```
01410
01411   return true;
01412 }
```

17.79 src/mtk_div_2d.cc File Reference

Implements the class Div2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_uni_stg_grid_1d.h"
#include "mtk_div_1d.h"
#include "mtk_div_2d.h"
```
Include dependency graph for mtk_div_2d.cc:

17.79.1 Detailed Description

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

Author

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

Definition in file mtk_div_2d.cc.

17.80 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
```

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
00069 mtk::Div2D::Div2D():
00070   order_accuracy_(),
00071   mimetic_threshold_() {}
00072
00073 mtk::Div2D::Div2D(const Div2D &div):
00074   order_accuracy_(div.order_accuracy_),
00075   mimetic_threshold_(div.mimetic_threshold_) {}
00076
00077 mtk::Div2D::~Div2D() {}
00078
00079 bool mtk::Div2D::ConstructDiv2D(const
00080   mtk::UniStgGrid2D &grid,
00080   int order_accuracy,
00081   mtk::Real mimetic_threshold) {
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 }
```

17.81 src/mtk_div_3d.cc File Reference

Implements the class Div3D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_div_1d.h"
#include "mtk_div_3d.h"
```

Include dependency graph for mtk_div_3d.cc:
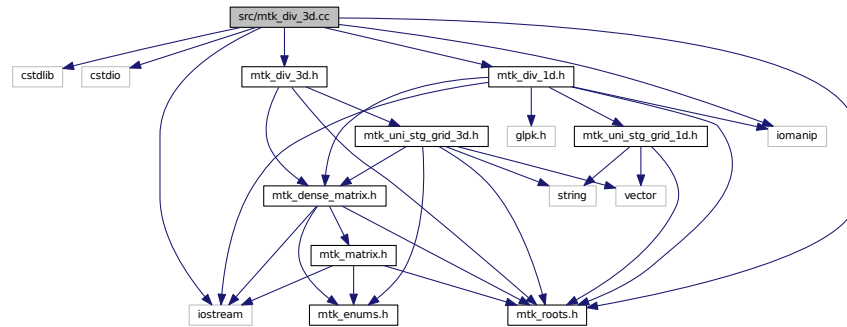
17.81.1 Detailed Description

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.

17.82 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_(),
00069   mimetic_threshold_() {}
00070
00071 mtk::Div3D::Div3D(const Div3D &grad):
00072   order_accuracy_(grad.order_accuracy_),
00073   mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Div3D::~Div3D() {}
00076
00077 bool mtk::Div3D::ConstructDiv3D(const
    mtk::UniStgGrid3D &grid,
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
00126
00127   mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(iz, iy));
00128   mtk::DenseMatrix dx(mtk::DenseMatrix::Kron(aux1, Dx));
00129
00131
00132   mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(iz, Dy));
00133   mtk::DenseMatrix dy(mtk::DenseMatrix::Kron(aux2, ix));
00134
00136
00137   mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Dz, iy));
00138   mtk::DenseMatrix dz(mtk::DenseMatrix::Kron(aux3, ix));
00139
00140   #if MTK_VERBOSE_LEVEL > 2
00141   std::cout << "Dx: " << mx << " by " << nx << std::endl;
00142   std::cout << "Ix: " << num_cells_x << " by " << nx  << std::endl;
00143   std::cout << "Dy: " << my << " by " << ny << std::endl;
00144   std::cout << "Iy: " << num_cells_y << " by " << ny  << std::endl;
00145   std::cout << "Dz: " << mz << " by " << nz << std::endl;
00146   std::cout << "Iz: " << num_cells_z << " by " << nz  << std::endl;
00147   #endif
00148
00150
00151   int total_rows{nx*ny*nz};
00152   int total_cols{mx*num_cells_y*num_cells_z +
00153                  num_cells_x*my*num_cells_z +
00154                  num_cells_x*num_cells_y*mz};
00155
00156   #if MTK_VERBOSE_LEVEL > 2
00157   std::cout << "Div 3D: " << total_rows << " by " << total_cols << std::endl;
00158   #endif
00159
00160   mtk::DenseMatrix d3d(total_rows, total_cols);
00161
00162   for (auto ii = 0; ii < total_rows; ++ii) {
00163
00164     for (auto jj = 0; jj < mx*num_cells_y*num_cells_z; ++jj) {
00165       d3d.SetValue(ii, jj, dx.GetValue(ii, jj));
00166     }
00167
00168     int offset = mx*num_cells_y*num_cells_z;
00169
00170     for(auto kk = 0; kk < num_cells_x*my*num_cells_z; ++kk) {
00171       d3d.SetValue(ii, kk + offset, dy.GetValue(ii, kk));
00172     }
00173
00174     offset += num_cells_x*my*num_cells_z;
00175
00176     for(auto ll = 0; ll < num_cells_x*num_cells_y*mz; ++ll) {
00177       d3d.SetValue(ii, ll + offset, dz.GetValue(ii, ll));
00178     }
00179   }
00180
00181   divergence_ = d3d;
00182
00183   return info;
00184 }
00185
00186 mtk::DenseMatrix mtk::Div3D::ReturnAsDenseMatrix() const {
00187
00188   return divergence_;
00189 }
```

## 17.83   src/mtk_glpk_adapter.cc File Reference

Adapter class for the GLPK API.

```
#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_glpk_adapter.h"
```
Include dependency graph for mtk_glpk_adapter.cc:



### 17.83.1 Detailed Description

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

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

**See also**

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

**Author**

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

Definition in file mtk_glpk_adapter.cc.

## 17.84 mtk_glpk_adapter.cc

```
00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
```

```
00064
00065 #include <cmath>
00066 #include <cstring>
00067
00068 #include <iostream>
00069 #include <iomanip>
00070 #include <limits>
00071
00072 #include "mtk_roots.h"
00073 #include "mtk_blas_adapter.h"
00074 #include "mtk_glpk_adapter.h"
00075
00076 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
       mtk::Real *A,
00077                                                    int nrows,
00078                                                    int ncols,
00079                                                    int kk,
00080                                                    mtk::Real *hh,
00081                                                    mtk::Real *qq,
00082                                                    int robjective,
00083                                                    mtk::Real mimetic_threshold,
00084                                                    int copy) {
00085
00086   #if MTK_DEBUG_LEVEL > 0
00087   char mps_file_name[18]; // File name for the MPS files.
00088   #endif
00089   char rname[5];          // Row name.
00090   char cname[5];          // Column name.
00091
00092   glp_prob *lp; // Linear programming problem.
00093
00094   int *ia;  // Array for the problem.
00095   int *ja;  // Array for the problem.
00096
00097   int problem_size; // Size of the problem.
00098   int lp_nrows;     // Number of rows.
00099   int lp_ncols;     // Number of columns.
00100   int matsize;      // Size of the matrix.
00101   int glp_index{1}; // Index of the objective function.
00102   int ii;           // Iterator.
00103   int jj;           // Iterator.
00104
00105   mtk::Real *ar;            // Array for the problem.
```

```
00106    mtk::Real *objective;      // Array containing the objective function.
00107    mtk::Real *rhs;            // Array containing the rhs.
00108    mtk::Real *err;            // Array of errors.
00109
00110    mtk::Real x1;              // Norm-2 of the error.
00111
00112    #if MTK_DEBUG_LEVEL > 0
00113    mtk::Real obj_value;       // Value of the objective function.
00114    #endif
00115
00116    lp_nrows = kk;
00117    lp_ncols = kk;
00118
00119    matsize = lp_nrows*lp_ncols;
00120
00122
00124    problem_size = lp_nrows*lp_ncols + 1;
00125
00126    try {
00127      ia = new int[problem_size];
00128    } catch (std::bad_alloc &memory_allocation_exception) {
00129      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00130        std::endl;
00131      std::cerr << memory_allocation_exception.what() << std::endl;
00132    }
00133    memset(ia, 0, sizeof(ia[0])*problem_size);
00134
00135    try {
00136      ja = new int[problem_size];
00137    } catch (std::bad_alloc &memory_allocation_exception) {
00138      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00139        std::endl;
00140      std::cerr << memory_allocation_exception.what() << std::endl;
00141    }
00142    memset(ja, 0, sizeof(ja[0])*problem_size);
00143
00144    try {
00145      ar = new mtk::Real[problem_size];
00146    } catch (std::bad_alloc &memory_allocation_exception) {
00147      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00148        std::endl;
00149      std::cerr << memory_allocation_exception.what() << std::endl;
00150    }
00151    memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00152
00153    try {
00154      objective = new mtk::Real[lp_ncols + 1];
00155    } catch (std::bad_alloc &memory_allocation_exception) {
00156      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00157        std::endl;
00158      std::cerr << memory_allocation_exception.what() << std::endl;
00159    }
00160    memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00161
00162    try {
00163      rhs = new mtk::Real[lp_nrows + 1];
00164    } catch (std::bad_alloc &memory_allocation_exception) {
00165      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00166        std::endl;
00167      std::cerr << memory_allocation_exception.what() << std::endl;
00168    }
00169    memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00170
00171    try {
00172      err = new mtk::Real[lp_nrows];
00173    } catch (std::bad_alloc &memory_allocation_exception) {
00174      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00175        std::endl;
00176      std::cerr << memory_allocation_exception.what() << std::endl;
00177    }
00178    memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00179
00180    #if MTK_DEBUG_LEVEL > 0
00181    std::cout << "Problem size: " << problem_size << std::endl;
00182    std::cout << "lp_nrows = " << lp_nrows << std::endl;
00183    std::cout << "lp_ncols = " << lp_ncols << std::endl;
00184    std::cout << std::endl;
00185    #endif
00186
00187    lp = glp_create_prob();
00188
```

```
00189    glp_set_prob_name (lp, "mtk::GLPKAdapter::Simplex");
00190
00191    glp_set_obj_dir (lp, GLP_MIN);
00192
00194
00195    glp_add_rows(lp, lp_nrows);
00196
00197    for (ii = 1; ii <= lp_nrows; ++ii) {
00198      sprintf(rname, "R%02d",ii);
00199      glp_set_row_name(lp, ii, rname);
00200    }
00201
00202    glp_add_cols(lp, lp_ncols);
00203
00204    for (ii = 1; ii <= lp_ncols; ++ii) {
00205      sprintf(cname, "Q%02d",ii);
00206      glp_set_col_name (lp, ii, cname);
00207    }
00208
00210
00211    #if MTK_DEBUG_LEVEL>0
00212    std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
00213    #endif
00214    for (jj = 0; jj < kk; ++jj) {
00215      objective[glp_index] = A[jj + robjective * ncols];
00216      glp_index++;
00217    }
00218    #if MTK_DEBUG_LEVEL >0
00219    std::cout << std::endl;
00220    #endif
00221
00223
00224    glp_index = 1;
00225    rhs[0] = mtk::kZero;
00226    for (ii = 0; ii <= lp_nrows; ++ii) {
00227      if (ii != robjective) {
00228        rhs[glp_index] = hh[ii];
00229        glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00230        glp_index++;
00231      }
00232    }
00233
00234    #if MTK_DEBUG_LEVEL > 0
00235    std::cout << "rhs =" << std::endl;
00236    for (auto ii = 0; ii < lp_nrows; ++ii) {
00237      std::cout << std::setw(15) << rhs[ii] << std::endl;
00238    }
00239    std::cout << std::endl;
00240    #endif
00241
00243
00244    for (ii = 1; ii <= lp_ncols; ++ii) {
00245      glp_set_obj_coef (lp, ii, objective[ii]);
00246    }
00247
00249
00250    for (ii = 1; ii <= lp_ncols; ++ii) {
00251      glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00252    }
00253
00255
00256    glp_index = 1;
00257    for (ii = 0; ii <= kk; ++ii) {
00258      for (jj = 0; jj < kk; ++jj) {
00259        if (ii != robjective) {
00260          ar[glp_index] = A[jj + ii * ncols];
00261          glp_index++;
00262        }
00263      }
00264    }
00265
00266    glp_index = 0;
00267
00268    for (ii = 1; ii < problem_size; ++ii) {
00269      if (((ii - 1) % lp_ncols) == 0) {
00270        glp_index++;
00271      }
00272      ia[ii] = glp_index;
00273      ja[ii] = (ii - 1) % lp_ncols + 1;
00274    }
00275
```

```
00276   glp_load_matrix (lp, matsize, ia, ja, ar);
00277
00278   #if MTK_DEBUG_LEVEL > 0
00279   sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00280   glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00281   #endif
00282
00284
00285   glp_simplex (lp, nullptr);
00286
00287   // Check status of the solution.
00288
00289   if (glp_get_status(lp) == GLP_OPT) {
00290
00291     for(ii = 1; ii <= lp_ncols; ++ii) {
00292       err[ii - 1] = qq[ii - 1] - glp_get_col_prim(lp,ii);
00293     }
00294
00295     #if MTK_DEBUG_LEVEL > 0
00296     obj_value = glp_get_obj_val (lp);
00297     std::cout << std::setw(12) << "CBS" << std::setw(12) << "CRS" << std::endl;
00298     for (ii = 0; ii < lp_ncols; ++ii) {
00299       std::cout << "q_" << ii + 1 << " = " << std::setw(12) <<
00300         glp_get_col_prim(lp,ii + 1) << std::setw(12) << qq[ii] << std::endl;
00301     }
00302     std::cout << "Objective function value (row " << robjective + 1 << ") = " <<
00303       obj_value << std::endl;
00304     #endif
00305
00306     if (copy) {
00307       for(ii = 0; ii < lp_ncols; ++ii) {
00308         qq[ii] = glp_get_col_prim(lp,ii + 1);
00309       }
00310       // Preserve the bottom values of qq.
00311     }
00312
00313     x1 = mtk::BLASAdapter::RealNRM2(err,lp_ncols);
00314
00315   } else {
00316     x1 = std::numeric_limits<mtk::Real>::infinity();
00317   }
00318
00319   glp_delete_prob (lp);
00320   glp_free_env ();
00321
00322   delete [] ia;
00323   delete [] ja;
00324   delete [] ar;
00325   delete [] objective;
00326   delete [] rhs;
00327   delete [] err;
00328
00329   return x1;
00330 }
```

17.85   src/mtk_grad_1d.cc File Reference

Implements the class Grad1D.

```
#include <cmath>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <limits>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_lapack_adapter.h"
#include "mtk_glpk_adapter.h"
#include "mtk_grad_1d.h"
```

Include dependency graph for mtk_grad_1d.cc:
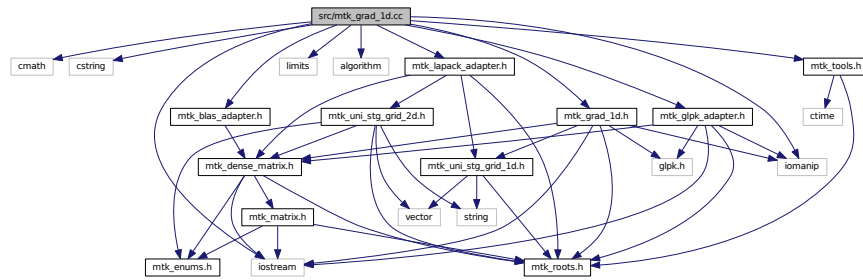
Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

Functions

- std::ostream & mtk::operator$<<$ (std::ostream &stream, mtk::Grad1D &in)

17.85.1 Detailed Description

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

Author

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

Todo Overload ostream operator as in mtk::Lap1D.

Todo Implement creation of ■ w. mtk::BLASAdapter.

Definition in file mtk_grad_1d.cc.

17.86 mtk_grad_1d.cc

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed, unless these modifications are made through the project's GitHub
00025 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00026 should be developed and included in any deliverable.
```

```
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions in binary form must reproduce the above copyright notice,
00032 this list of conditions and the following disclaimer in the documentation and/or
00033 other materials provided with the distribution.
00034
00035 4. Usage of the binary form on proprietary applications shall require explicit
00036 prior written permission from the the copyright holders, and due credit should
00037 be given to the copyright holders.
00038
00039 5. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066 #include <limits>
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074
00075 #include "mtk_grad_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &in) {
00080
00082
00083   stream << "gradient_[0] = " << std::setw(9) << in.gradient_[0] << std::endl;
00084
00086
00087   stream << "gradient_[1:" << in.order_accuracy_ << "] = ";
00088   for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00089     stream << std::setw(9) << in.gradient_[ii] << " ";
00090   }
00091   stream << std::endl;
00092
00094
00095   stream << "gradient_[" << in.order_accuracy_ + 1 << ":" <<
00096     2*in.order_accuracy_ << "] = ";
00097   for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
   order_accuracy_; ++ii) {
00098     stream << std::setw(9) << in.gradient_[ii] << " ";
00099   }
00100   stream << std::endl;
00101
00103
00104   int offset{2*in.order_accuracy_ + 1};
00105   int mm {};
00106
00107   stream << "gradient_[" << offset + mm << ":" <<
00108     offset + mm + in.num_bndy_coeffs_ - 1 << "] = ";
00109
00110   if (in.order_accuracy_ > mtk::kDefaultOrderAccuracy) {
```

```
00111      for (auto ii = 0; ii < in.num_bndy_approxs_ ; ++ii) {
00112        for (auto jj = 0; jj < in.num_bndy_coeffs_; jj++) {
00113          auto value = in.gradient_[offset + (mm)];
00114          stream << std::setw(9) << value << " ";
00115          mm++;
00116        }
00117      }
00118    } else {
00119      stream << std::setw(9) << in.gradient_[offset + 0] << ' ';
00120      stream << std::setw(9) << in.gradient_[offset + 1] << ' ';
00121      stream << std::setw(9) << in.gradient_[offset + 2] << ' ';
00122    }
00123    stream << std::endl;
00124
00125    return stream;
00126 }
00127 }
00128
00129 mtk::Grad1D::Grad1D():
00130    order_accuracy_(mtk::kDefaultOrderAccuracy),
00131    dim_null_(),
00132    num_bndy_approxs_(),
00133    num_bndy_coeffs_(),
00134    gradient_length_(),
00135    minrow_(),
00136    row_(),
00137    coeffs_interior_(),
00138    prem_apps_(),
00139    weights_crs_(),
00140    weights_cbs_(),
00141    mim_bndy_(),
00142    gradient_(),
00143    mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00144
00145 mtk::Grad1D::Grad1D(const Grad1D &grad):
00146    order_accuracy_(grad.order_accuracy_),
00147    dim_null_(grad.dim_null_),
00148    num_bndy_approxs_(grad.num_bndy_approxs_),
00149    num_bndy_coeffs_(grad.num_bndy_coeffs_),
00150    gradient_length_(grad.gradient_length_),
00151    minrow_(grad.minrow_),
00152    row_(grad.row_),
00153    coeffs_interior_(grad.coeffs_interior_),
00154    prem_apps_(grad.prem_apps_),
00155    weights_crs_(grad.weights_crs_),
00156    weights_cbs_(grad.weights_cbs_),
00157    mim_bndy_(grad.mim_bndy_),
00158    gradient_(grad.gradient_),
00159    mimetic_threshold_(grad.mimetic_threshold_) {}
00160
00161 mtk::Grad1D::~Grad1D() {
00162
00163    delete[] coeffs_interior_;
00164    coeffs_interior_ = nullptr;
00165
00166    delete[] prem_apps_;
00167    prem_apps_ = nullptr;
00168
00169    delete[] weights_crs_;
00170    weights_crs_ = nullptr;
00171
00172    delete[] weights_cbs_;
00173    weights_cbs_ = nullptr;
00174
00175    delete[] mim_bndy_;
00176    mim_bndy_ = nullptr;
00177
00178    delete[] gradient_;
00179    gradient_ = nullptr;
00180 }
00181
00182 bool mtk::Grad1D::ConstructGrad1D(int order_accuracy,
00183    Real mimetic_threshold) {
00184
00185    #ifdef MTK_PERFORM_PREVENTIONS
00185    mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00186    mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00187    mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00188                        __FILE__, __LINE__, __func__);
00189
00190    if (order_accuracy >= mtk::kCriticalOrderAccuracyGrad) {
```

```
00191    std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00192  }
00193
00194  std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00195  std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00196  #endif
00197
00198  order_accuracy_ = order_accuracy;
00199  mimetic_threshold_ = mimetic_threshold;
00200
00202  bool abort_construction = ComputeStencilInteriorGrid();
00203
00204  #ifdef MTK_PERFORM_PREVENTIONS
00205  if (!abort_construction) {
00206    std::cerr << "Could NOT complete stage 1." << std::endl;
00207    std::cerr << "Exiting..." << std::endl;
00208    return false;
00209  }
00210  #endif
00211
00212  // At this point, we already have the values for the interior stencil stored
00213  // in the coeffs_interior_ array.
00214
00215  dim_null_ = order_accuracy_/2 - 1;
00216
00217  num_bndy_approxs_ = dim_null_ + 1;
00218
00219  #ifdef MTK_PRECISION_DOUBLE
00220  num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00221  #else
00222  num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00223  #endif
00224
00226
00227  // For this we will follow recommendations given in:
00228  //
00229  // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00230  //
00231  // We will compute the QR Factorization of the transpose, as in the
00232  // following (MATLAB) pseudo-code:
00233  //
00234  // [Q,R] = qr(V'); % Full QR as defined in
00235  // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00236  //
00237  // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00238  //
00239  // However, given the nature of the Vandermonde matrices we've just
00240  // computed, they all posses the same null-space. Therefore, we impose the
00241  // convention of computing the null-space of the first Vandermonde matrix
00242  // (west boundary).
00243
00244  // In the case of the gradient, the first Vandermonde system has a unique
00245  // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00246  // matrix used to assemble said system, will have an empty null-space.
00247
00248  // Therefore, we only compute a rational basis for the case of order higher
00249  // than second.
00250
00251  if (dim_null_ > 0) {
00252
00253    abort_construction = ComputeRationalBasisNullSpace();
00254
00255    #ifdef MTK_PERFORM_PREVENTIONS
00256    if (!abort_construction) {
00257      std::cerr << "Could NOT complete stage 2.1." << std::endl;
00258      std::cerr << "Exiting..." << std::endl;
00259      return false;
00260    }
00261    #endif
00262  }
00263
00265  abort_construction = ComputePreliminaryApproximations();
00266
00267  #ifdef MTK_PERFORM_PREVENTIONS
00268  if (!abort_construction) {
00269    std::cerr << "Could NOT complete stage 2.2." << std::endl;
00270    std::cerr << "Exiting..." << std::endl;
00271    return false;
00272  }
00273  #endif
00274
```

```
00276    abort_construction = ComputeWeights();
00277
00278    #ifdef MTK_PERFORM_PREVENTIONS
00279    if (!abort_construction) {
00280      std::cerr << "Could NOT complete stage 2.3." << std::endl;
00281      std::cerr << "Exiting..." << std::endl;
00282      return false;
00283    }
00284    #endif
00285
00287    if (dim_null_ > 0) {
00288
00289      abort_construction = ComputeStencilBoundaryGrid();
00290
00291      #ifdef MTK_PERFORM_PREVENTIONS
00292      if (!abort_construction) {
00293        std::cerr << "Could NOT complete stage 2.4." << std::endl;
00294        std::cerr << "Exiting..." << std::endl;
00295        return false;
00296      }
00297      #endif
00298    }
00299

00301    // Once we have the following three collections of data:
00302    //   (a) the coefficients for the interior,
00303    //   (b)  the coefficients for the boundary (if it applies),
00304    //   (c)  and the weights (if it applies),
00305    // we will store everything in the output array:
00306
00307
00308    abort_construction = AssembleOperator();
00309
00310    #ifdef MTK_PERFORM_PREVENTIONS
00311    if (!abort_construction) {
00312      std::cerr << "Could NOT complete stage 3." << std::endl;
00313      std::cerr << "Exiting..." << std::endl;
00314      return false;
00315    }
00316    #endif
00317
00318    return true;
00319 }
00320
00321 int mtk::Grad1D::num_bndy_coeffs() const {
00322
00323    return num_bndy_coeffs_;
00324 }
00325
00326 mtk::Real *mtk::Grad1D::coeffs_interior() const {
00327
00328    return coeffs_interior_;
00329 }
00330
00331 mtk::Real *mtk::Grad1D::weights_crs() const {
00332
00333    return weights_crs_;
00334 }
00335
00336 mtk::Real *mtk::Grad1D::weights_cbs() const {
00337
00338    return weights_cbs_;
00339 }
00340
00341 mtk::DenseMatrix mtk::Grad1D::mim_bndy() const {
00342
00343    mtk::DenseMatrix xx(dim_null_ + 1, 3*order_accuracy_/2);
00344
00345    auto counter = 0;
00346    for (auto ii = 0; ii < dim_null_ + 1; ++ii) {
00347      for(auto jj = 0; jj < 3*order_accuracy_/2; ++jj) {
00348        xx.SetValue(ii,jj, gradient_[2*order_accuracy_ + 1 + counter]);
00349        counter++;
00350      }
00351    }
00352
00353    return xx;
00354 }
00355
00356 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00357    mtk::Real west,
                                                    mtk::Real east,
```

```
00358                                                     int num_cells_x) const {
00359
00360    int nn{num_cells_x}; // Number of cells on the grid.
00361
00362    #ifdef MTK_PERFORM_PREVENTIONS
00363    mtk::Tools::Prevent(east < west, __FILE__, __LINE__, __func__);
00364    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00365    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00366    #endif
00367
00368    mtk::Real delta_x = (east - west)/((mtk::Real) num_cells_x);
00369
00370    mtk::Real inv_delta_x{mtk::kOne/delta_x};
00371
00372    int gg_num_rows = nn + 1;
00373    int gg_num_cols = nn + 2;
00374    int elements_per_row = num_bndy_coeffs_;
00375    int num_extra_rows = order_accuracy_/2;
00376
00377    // Output matrix featuring sizes for gradient operators.
00378    mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00379
00381
00382    auto ee_index = 0;
00383    for (auto ii = 0; ii < num_extra_rows; ii++) {
00384      auto cc = 0;
00385      for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00386        if(cc >= elements_per_row) {
00387          out.SetValue(ii, jj, mtk::kZero);
00388        } else {
00389          out.SetValue(ii,jj,
00390                       gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00391          cc++;
00392        }
00393      }
00394    }
00395
00397
00398    for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00399      auto jj = ii - num_extra_rows + 1;
00400      for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00401        out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00402      }
00403    }
00404
00406
00407    ee_index = 0;
00408    for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00409      auto cc = 0;
00410      for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00411        if(cc >= elements_per_row) {
00412          out.SetValue(ii,jj,mtk::kZero);
00413        } else {
00414          out.SetValue(ii,jj,
00415                       -gradient_[2*order_accuracy_ + 1 +
00416 ee_index++]*inv_delta_x);
00417          cc++;
00418        }
00419      }
00420    }
00421
00422    return out;
00423 }
00424
00425 mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(
00426    const UniStgGrid1D &grid) const {
00427
00428    int nn{grid.num_cells_x()}; // Number of cells on the grid.
00429
00430    #ifdef MTK_PERFORM_PREVENTIONS
00431    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00432    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00433    #endif
00434
00435    mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00436
00437    int gg_num_rows = nn + 1;
00438    int gg_num_cols = nn + 2;
00439    int elements_per_row = num_bndy_coeffs_;
00440    int num_extra_rows = order_accuracy_/2;
00441
```

```
00442    // Output matrix featuring sizes for gradient operators.
00443    mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00444
00446
00447    auto ee_index = 0;
00448    for (auto ii = 0; ii < num_extra_rows; ii++) {
00449      auto cc = 0;
00450      for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00451        if(cc >= elements_per_row) {
00452          out.SetValue(ii, jj, mtk::kZero);
00453        } else {
00454          out.SetValue(ii,jj,
00455                       gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00456          cc++;
00457        }
00458      }
00459    }
00460
00462
00463    for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00464      auto jj = ii - num_extra_rows + 1;
00465      for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00466        out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00467      }
00468    }
00469
00471
00472    ee_index = 0;
00473    for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00474      auto cc = 0;
00475      for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00476        if(cc >= elements_per_row) {
00477          out.SetValue(ii,jj,mtk::kZero);
00478        } else {
00479          out.SetValue(ii,jj,
00480                       -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00481          cc++;
00482        }
00483      }
00484    }
00485
00486    return out;
00487  }
00488
00489  mtk::DenseMatrix mtk::Grad1D::ReturnAsDimensionlessDenseMatrix
        (
00490    int num_cells_x) const {
00491
00492    int nn{num_cells_x}; // Number of cells on the grid.
00493
00494    #ifdef MTK_PERFORM_PREVENTIONS
00495    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00496    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00497    #endif
00498
00499    int gg_num_rows = nn + 1;
00500    int gg_num_cols = nn + 2;
00501    int elements_per_row = num_bndy_coeffs_;
00502    int num_extra_rows = order_accuracy_/2;
00503
00504    // Output matrix featuring sizes for gradient operators.
00505    mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00506
00508
00509    auto ee_index = 0;
00510    for (auto ii = 0; ii < num_extra_rows; ii++) {
00511      auto cc = 0;
00512      for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00513        if(cc >= elements_per_row) {
00514          out.SetValue(ii, jj, mtk::kZero);
00515        } else {
00516          out.SetValue(ii,jj,
00517                       gradient_[2*order_accuracy_ + 1 + ee_index++]);
00518          cc++;
00519        }
00520      }
00521    }
00522
00524
00525    for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00526      auto jj = ii - num_extra_rows + 1;
```

```
00527       for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00528         out.SetValue(ii, jj, coeffs_interior_[cc]);
00529       }
00530     }
00531
00533
00534     ee_index = 0;
00535     for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00536       auto cc = 0;
00537       for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00538         if(cc >= elements_per_row) {
00539           out.SetValue(ii,jj,mtk::kZero);
00540         } else {
00541           out.SetValue(ii,jj,
00542                       -gradient_[2*order_accuracy_ + 1 + ee_index++]);
00543           cc++;
00544         }
00545       }
00546     }
00547
00548     return out;
00549 }
00550
00551 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00552
00554
00555     mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00556
00557     try {
00558       pp = new mtk::Real[order_accuracy_];
00559     } catch (std::bad_alloc &memory_allocation_exception) {
00560       std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00561         std::endl;
00562       std::cerr << memory_allocation_exception.what() << std::endl;
00563     }
00564     memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00565
00566     #ifdef MTK_PRECISION_DOUBLE
00567     pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00568     #else
00569     pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00570     #endif
00571
00572     for (auto ii = 1; ii < order_accuracy_; ++ii) {
00573       pp[ii] = pp[ii - 1] + mtk::kOne;
00574     }
00575
00576     #if MTK_VERBOSE_LEVEL > 3
00577     std::cout << "pp =" << std::endl;
00578     for (auto ii = 0; ii < order_accuracy_; ++ii) {
00579       std::cout << std::setw(12) << pp[ii];
00580     }
00581     std::cout << std::endl << std::endl;
00582     #endif
00583
00585
00586     bool transpose{false};
00587
00588     mtk::DenseMatrix vander_matrix(pp,order_accuracy_,order_accuracy_,transpose);
00589
00590     #if MTK_VERBOSE_LEVEL > 4
00591     std::cout << "vander_matrix = " << std::endl;
00592     std::cout << vander_matrix << std::endl << std::endl;
00593     #endif
00594
00596
00597     try {
00598       coeffs_interior_ = new mtk::Real[order_accuracy_];
00599     } catch (std::bad_alloc &memory_allocation_exception) {
00600       std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00601         std::endl;
00602       std::cerr << memory_allocation_exception.what() << std::endl;
00603     }
00604     memset(coeffs_interior_, mtk::kZero,
00605 sizeof(coeffs_interior_[0])*order_accuracy_);
00606
00607     coeffs_interior_[1] = mtk::kOne;
00608
00609     #if MTK_VERBOSE_LEVEL > 3
00610     std::cout << "oo =" << std::endl;
00611     for (auto ii = 0; ii < order_accuracy_; ++ii) {
```

```
00612      std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00613    }
00614    std::cout << std::endl;
00615    #endif
00616
00618
00619    int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00620                                                  coeffs_interior_)};
00621
00622    #ifdef MTK_PERFORM_PREVENTIONS
00623    if (!info) {
00624      std::cout << "System solved! Interior stencil attained!" << std::endl;
00625      std::cout << std::endl;
00626    }
00627    else {
00628      std::cerr << "Something wrong solving system! info = " << info << std::endl;
00629      std::cerr << "Exiting..." << std::endl;
00630      return false;
00631    }
00632    #endif
00633
00634    #if MTK_VERBOSE_LEVEL > 3
00635    std::cout << "coeffs_interior_ =" << std::endl;
00636    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00637      std::cout << std::setw(12) << coeffs_interior_[ii];
00638    }
00639    std::cout << std::endl << std::endl;
00640    #endif
00641
00642    delete [] pp;
00643    pp = nullptr;
00644
00645    return true;
00646  }
00647
00648  bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00649
00651
00652    mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00653
00654    try {
00655      gg = new mtk::Real[num_bndy_coeffs_];
00656    } catch (std::bad_alloc &memory_allocation_exception) {
00657      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00658        std::endl;
00659      std::cerr << memory_allocation_exception.what() << std::endl;
00660    }
00661    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00662
00663    #ifdef MTK_PRECISION_DOUBLE
00664    gg[1] = 1.0/2.0;
00665    #else
00666    gg[1] = 1.0f/2.0f;
00667    #endif
00668    for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00669      gg[ii] = gg[ii - 1] + mtk::kOne;
00670    }
00671
00672    #if MTK_VERBOSE_LEVEL > 3
00673    std::cout << "gg =" << std::endl;
00674    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00675      std::cout << std::setw(12) << gg[ii];
00676    }
00677    std::cout << std::endl << std::endl;
00678    #endif
00679
00681
00682    bool tran{true}; // Should I transpose the Vandermonde matrix.
00683
00684    mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00685
00686    #if MTK_VERBOSE_LEVEL > 4
00687    std::cout << "aa_west_t =" << std::endl;
00688    std::cout << aa_west_t << std::endl;
00689    #endif
00690
00692
00693    mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
00      (aa_west_t));
00694
00695    #if MTK_VERBOSE_LEVEL > 3
```

```
00696    std::cout << "qq_t = " << std::endl;
00697    std::cout << qq_t << std::endl;
00698    #endif
00699
00701
00702    int kk_num_rows{num_bndy_coeffs_};
00703    int kk_num_cols{dim_null_};
00704
00705    mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00706
00707    // In the case of the gradient, even though we must solve for a null-space
00708    // of dimension 2, we must only extract ONE basis for the kernel.
00709    // We perform this extraction here:
00710
00711    int aux_{kk_num_rows - kk_num_cols};
00712    for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00713      aux_--;
00714      for (auto jj = 0; jj < kk_num_rows; jj++) {
00715        kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =
00716          qq_t.data()[ii*num_bndy_coeffs_ + jj];
00717      }
00718    }
00719
00720    #if MTK_VERBOSE_LEVEL > 2
00721    std::cout << "kk =" << std::endl;
00722    std::cout << kk << std::endl;
00723    std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00724    std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00725    std::cout << std::endl;
00726    #endif
00727
00729
00730    // Scale thus requesting that the last entries of the attained basis for the
00731    // null-space, adopt the pattern we require.
00732    // Essentially we will implement the following MATLAB pseudo-code:
00733    //  scalers = kk(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00734    //  SK = kk*scalers
00735    // where SK is the scaled null-space.
00736
00737    // In this point, we almost have all the data we need correctly allocated
00738    // in memory. We will create the matrix iden_, and elements we wish to scale
00739    // in the kk array. Using the concept of the leading dimension, we could just
00740    // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00741    // GET how does it work. So I will just create a matrix with the content of
00742    // this array that we need, solve for the scalers and then scale the
00743    // whole kk:
00744
00745    // We will then create memory for that sub-matrix of kk (subk).
00746
00747    mtk::DenseMatrix subk(dim_null_, dim_null_);
00748
00749    auto zz = 0;
00750    for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00751      for (auto jj = 0; jj < dim_null_; jj++) {
00752        subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00753      }
00754      zz++;
00755    }
00756
00757    #if MTK_VERBOSE_LEVEL > 4
00758    std::cout << "subk =" << std::endl;
00759    std::cout << subk << std::endl;
00760    #endif
00761
00762    subk.Transpose();
00763
00764    #if MTK_VERBOSE_LEVEL > 4
00765    std::cout << "subk_t =" << std::endl;
00766    std::cout << subk << std::endl;
00767    #endif
00768
00769    bool padded{false};
00770    tran = false;
00771
00772    mtk::DenseMatrix iden(dim_null_, padded, tran);
00773
00774    #if MTK_VERBOSE_LEVEL > 4
00775    std::cout << "iden =" << std::endl;
00776    std::cout << iden << std::endl;
00777    #endif
00778
```

```
00779    // Solve the system to compute the scalers.
00780    // An example of the system to solve, for k = 8, is:
00781    //
00782    // subk*scalers = iden or
00783    //
00784    // |  0.386018  -0.0339244   -0.129478 |           | 1 0 0 |
00785    // | -0.119774   0.0199423   0.0558632 |*scalers = | 0 1 0 |
00786    // | 0.0155708 -0.00349546 -0.00853182 |           | 0 0 1 |
00787    //
00788    // Notice this is a nrhs = 3 system.
00789    // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00790    // will be stored in the created identity matrix.
00791    // Let us first transpose subk (because of LAPACK):
00792
00793    int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00794
00795    #ifdef MTK_PERFORM_PREVENTIONS
00796    if (!info) {
00797      std::cout << "System successfully solved!" <<
00798        std::endl;
00799    } else {
00800      std::cerr << "Something went wrong solving system! info = " << info <<
00801        std::endl;
00802      std::cerr << "Exiting..." << std::endl;
00803      return false;
00804    }
00805    std::cout << std::endl;
00806    #endif
00807
00808    #if MTK_VERBOSE_LEVEL > 4
00809    std::cout << "Computed scalers:" << std::endl;
00810    std::cout << iden << std::endl;
00811    #endif
00812
00813    // Multiply the two matrices to attain a scaled basis for null-space.
00814
00815    rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00816
00817    #if MTK_VERBOSE_LEVEL > 4
00818    std::cout << "Rational basis for the null-space:" << std::endl;
00819    std::cout << rat_basis_null_space_ << std::endl;
00820    #endif
00821
00822    // At this point, we have a rational basis for the null-space, with the
00823    // pattern we need! :)
00824
00825    delete [] gg;
00826    gg = nullptr;
00827
00828    return true;
00829 }
00830
00831 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00832
00834
00835    mtk::Real *gg{}; // Generator vector for the first approximation.
00836
00837    try {
00838      gg = new mtk::Real[num_bndy_coeffs_];
00839    } catch (std::bad_alloc &memory_allocation_exception) {
00840      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00841        std::endl;
00842      std::cerr << memory_allocation_exception.what() << std::endl;
00843    }
00844    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00845
00846    #ifdef MTK_PRECISION_DOUBLE
00847    gg[1] = 1.0/2.0;
00848    #else
00849    gg[1] = 1.0f/2.0f;
00850    #endif
00851    for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00852      gg[ii] = gg[ii - 1] + mtk::kOne;
00853    }
00854
00855    #if MTK_VERBOSE_LEVEL > 3
00856    std::cout << "gg0 =" << std::endl;
00857    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00858      std::cout << std::setw(12) << gg[ii];
00859    }
00860    std::cout << std::endl << std::endl;
```

```
00861    #endif
00862
00863    // Allocate 2D array to store the collection of preliminary approximations.
00864    try {
00865      prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00866    } catch (std::bad_alloc &memory_allocation_exception) {
00867      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00868 std::endl;
00869      std::cerr << memory_allocation_exception.what() << std::endl;
00870    }
00871    memset(prem_apps_,
00872           mtk::kZero,
00873           sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00874
00875
00876
00877    for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00878
00879      // Re-check new generator vector for every iteration except for the first.
00880      #if MTK_VERBOSE_LEVEL > 3
00881      if (ll > 0) {
00882        std::cout << "gg_" << ll << " =" << std::endl;
00883        for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00884          std::cout << std::setw(12) << gg[ii];
00885        }
00886        std::cout << std::endl << std::endl;
00887      }
00888      #endif
00889
00890
00891
00892      bool transpose{false};
00893
00894      mtk::DenseMatrix aa(gg,
00895                          num_bndy_coeffs_, order_accuracy_ + 1,
00896                          transpose);
00897
00898      #if MTK_VERBOSE_LEVEL > 4
00899      std::cout << "aa_" << ll << " =" << std::endl;
00900      std::cout << aa << std::endl;
00901      #endif
00902
00903
00904      mtk::Real *ob{};
00905
00906
00907      auto ob_ld = num_bndy_coeffs_;
00908
00909      try {
00910        ob = new mtk::Real[ob_ld];
00911      } catch (std::bad_alloc &memory_allocation_exception) {
00912        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00913          std::endl;
00914        std::cerr << memory_allocation_exception.what() << std::endl;
00915      }
00916      memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00917
00918      ob[1] = mtk::kOne;
00919
00920      #if MTK_VERBOSE_LEVEL > 3
00921      std::cout << "ob = " << std::endl << std::endl;
00922      for (auto ii = 0; ii < ob_ld; ++ii) {
00923        std::cout << std::setw(12) << ob[ii] << std::endl;
00924      }
00925      std::cout << std::endl;
00926      #endif
00927
00928
00929      // However, this is an under-determined system of equations. So we can not
00930      // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00931      // our LAPACKAdapter class.
00932
00933
00934      int info_{
00935        mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
00936   , ob_ld)};
00937
00938      #ifdef MTK_PERFORM_PREVENTIONS
00939      if (!info_) {
00940        std::cout << "System successfully solved!" << std::endl << std::endl;
00941      } else {
00942        std::cerr << "Error solving system! info = " << info_ << std::endl;
00943        return false;
00944      }
00945      #endif
```

```
00945
00946      #if MTK_VERBOSE_LEVEL > 3
00947      std::cout << "ob =" << std::endl;
00948      for (auto ii = 0; ii < ob_ld; ++ii) {
00949        std::cout << std::setw(12) << ob[ii] << std::endl;
00950      }
00951      std::cout << std::endl;
00952      #endif
00953
00955
00956      // This implies a DAXPY operation. However, we must construct the arguments
00957      // for this operation.
00958
00960      // Save them into the ob_bottom array:
00961
00962      Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00963
00964      try {
00965        ob_bottom = new mtk::Real[dim_null_];
00966      } catch (std::bad_alloc &memory_allocation_exception) {
00967        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00968          std::endl;
00969        std::cerr << memory_allocation_exception.what() << std::endl;
00970      }
00971      memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00972
00973      for (auto ii = 0; ii < dim_null_; ++ii) {
00974        ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00975      }
00976
00977      #if MTK_VERBOSE_LEVEL > 3
00978      std::cout << "ob_bottom =" << std::endl;
00979      for (auto ii = 0; ii < dim_null_; ++ii) {
00980        std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00981      }
00982      std::cout << std::endl;
00983      #endif
00984
00986
00987      // We must computed an scaled ob, sob, using the scaled null-space in
00988      // rat_basis_null_space_.
00989      // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00990      // or:                ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00991      // thus:              Y =    a*A     *x          +    b*Y (DAXPY).
00992
00993      #if MTK_VERBOSE_LEVEL > 4
00994      std::cout << "Rational basis for the null-space:" << std::endl;
00995      std::cout << rat_basis_null_space_ << std::endl;
00996      #endif
00997
00998      mtk::Real alpha{-mtk::kOne};
00999      mtk::Real beta{mtk::kOne};
01000
01001      mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
01002                                    ob_bottom, beta, ob);
01003
01004      #if MTK_VERBOSE_LEVEL > 3
01005      std::cout << "scaled ob:" << std::endl;
01006      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01007        std::cout << std::setw(12) << ob[ii] << std::endl;
01008      }
01009      std::cout << std::endl;
01010      #endif
01011
01012      // We save the recently scaled solution, into an array containing these.
01013      // We can NOT start building the pi matrix, simply because I want that part
01014      // to be separated since its construction depends on the algorithm we want
01015      // to implement.
01016
01017      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01018        prem_apps_[ii*num_bndy_approxs_ + ll] = ob[ii];
01019      }
01020
01021      // After the first iteration, simply shift the entries of the last
01022      // generator vector used:
01023      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01024        gg[ii]--;
01025      }
01026
01027      // Garbage collection for this loop:
01028      delete[] ob;
```

```
01029    ob = nullptr;
01030
01031    delete[] ob_bottom;
01032    ob_bottom = nullptr;
01033  } // End of: for (ll = 0; ll < dim_null; ll++);
01034
01035  #if MTK_VERBOSE_LEVEL > 4
01036  std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
01037  for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01038    for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01039      std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
01040    }
01041    std::cout << std::endl;
01042  }
01043  std::cout << std::endl;
01044  #endif
01045
01046  delete[] gg;
01047  gg = nullptr;
01048
01049  return true;
01050 }
01051
01052 bool mtk::Grad1D::ComputeWeights() {
01053
01054  // Matrix to compute the weights as in the CRSA.
01055  mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
01056
01057
01058
01059  // Assemble the pi matrix using:
01060  // 1. The collection of scaled preliminary approximations.
01061  // 2. The collection of coefficients approximating at the interior.
01062  // 3. The scaled basis for the null-space.
01063
01064  // 1.1. Process array of scaled preliminary approximations.
01065
01066  // These are queued in scaled_solutions. Each one of these, will be a column
01067  // of the pi matrix:
01068  for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01069    for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01070      pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
01071        prem_apps_[ii*num_bndy_approxs_ + jj];
01072    }
01073  }
01074
01075  // 1.2. Add columns from known stencil approximating at the interior.
01076
01077  // However, these must be padded by zeros, according to their position in the
01078  // final pi matrix:
01079  auto mm = 1;
01080  for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
01081    for (auto ii = 0; ii < order_accuracy_; ++ii) {
01082      auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
01083        (order_accuracy_/2 + 1)) + jj;
01084      pi.data()[de] = coeffs_interior_[ii];
01085    }
01086    ++mm;
01087  }
01088
01089  rat_basis_null_space_.OrderColMajor();
01090
01091  #if MTK_VERBOSE_LEVEL > 4
01092  std::cout << "Rational basis for the null-space (col. major):" << std::endl;
01093  std::cout << rat_basis_null_space_ << std::endl;
01094  #endif
01095
01096  // 1.3. Add final set of columns: rational basis for null-space.
01097
01098  for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
01099       jj < num_bndy_coeffs_ - 1; ++jj) {
01100    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01101      auto og =
01102        (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
01103      auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
01104      pi.data()[de] = rat_basis_null_space_.data()[og];
01105    }
01106  }
01107
01108  #if MTK_VERBOSE_LEVEL > 4
01109  std::cout << "coeffs_interior_ =" << std::endl;
01110  for (auto ii = 0; ii < order_accuracy_; ++ii) {
```

```
01111      std::cout << std::setw(12) << coeffs_interior_[ii];
01112    }
01113    std::cout << std::endl << std::endl;
01114    #endif
01115
01116    #if MTK_VERBOSE_LEVEL > 4
01117    std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
01118    std::cout << pi << std::endl;
01119    #endif
01120
01121
01122
01123    // This imposes the mimetic condition.
01124
01125    mtk::Real *hh{};  // Right-hand side to compute weights in the C{R,B}SA.
01126
01127    try {
01128      hh = new mtk::Real[num_bndy_coeffs_];
01129    } catch (std::bad_alloc &memory_allocation_exception) {
01130      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01131        std::endl;
01132      std::cerr << memory_allocation_exception.what() << std::endl;
01133    }
01134    memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
01135
01136    hh[0] = -mtk::kOne;
01137    for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
01138      auto aux_xx = mtk::kZero;
01139      for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
01140        aux_xx += coeffs_interior_[jj];
01141      }
01142      hh[ii] = -mtk::kOne*aux_xx;
01143    }
01144
01146    // That is, we construct a system, to solve for the weights.
01147
01148
01149    // Once again we face the challenge of solving with LAPACK. However, for the
01150    // CRSA, this matrix PI is over-determined, since it has more rows than
01151    // unknowns. However, according to the theory, the solution to this system is
01152    // unique. We will use dgels_.
01153
01154    try {
01155      weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01156    } catch (std::bad_alloc &memory_allocation_exception) {
01157      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01158        std::endl;
01159      std::cerr << memory_allocation_exception.what() << std::endl;
01160    }
01161    memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01162
01163    int weights_ld{pi.num_cols() + 1};
01164
01165    // Preserve hh.
01166    std::copy(hh, hh + weights_ld, weights_cbs_);
01167
01168    pi.Transpose();
01169
01170    int info{
01171      mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01172                                                      weights_cbs_, weights_ld)
01173    };
01174
01175    #ifdef MTK_PERFORM_PREVENTIONS
01176    if (!info) {
01177      std::cout << "System successfully solved!" << std::endl << std::endl;
01178    } else {
01179      std::cerr << "Error solving system! info = " << info << std::endl;
01180      return false;
01181    }
01182    #endif
01183
01184    #if MTK_VERBOSE_LEVEL > 3
01185    std::cout << "hh =" << std::endl;
01186    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01187      std::cout << std::setw(11) << hh[ii] << std::endl;
01188    }
01189    std::cout << std::endl;
01190    #endif
01191
01192    // Preserve the original weights for research.
01193
```

```
01194    try {
01195      weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01196    } catch (std::bad_alloc &memory_allocation_exception) {
01197      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01198        std::endl;
01199      std::cerr << memory_allocation_exception.what() << std::endl;
01200    }
01201    memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01202
01203    std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01204
01205    #if MTK_VERBOSE_LEVEL > 3
01206    std::cout << "weights_CRSA + lambda =" << std::endl;
01207    for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01208      std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01209    }
01210    std::cout << std::endl;
01211    #endif
01212
01214
01215    if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01216
01217      int minrow_{std::numeric_limits<int>::infinity()};
01218
01219      mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
      order_accuracy_)};
01220      mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01221
01223
01224      mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01225
01226      // 6.1. Insert preliminary approximations to first set of columns.
01227
01228      for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01229        for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01230          phi.data()[ii*(order_accuracy_) + jj] =
01231            prem_apps_[ii*num_bndy_approxs_ + jj];
01232        }
01233      }
01234
01235      // 6.2. Skip a column and negate preliminary approximations.
01236
01237      for (auto jj = 0; jj < order_accuracy_ + 1; jj++) {
01238        for (auto ii = 1; ii < num_bndy_approxs_; ii++) {
01239          auto de = (ii+ order_accuracy_ - num_bndy_approxs_+ jj*order_accuracy_);
01240          auto og = (num_bndy_approxs_ - ii + (jj)*num_bndy_approxs_);
01241          phi.data()[de] = -prem_apps_[og];
01242        }
01243      }
01244
01245      // 6.3. Flip negative columns up-down.
01246
01247      for (auto ii = 0; ii < order_accuracy_/2; ii++) {
01248        for (auto jj = num_bndy_approxs_ + 1; jj < order_accuracy_; jj++) {
01249          auto aux = phi.data()[ii*order_accuracy_ + jj];
01250          phi.data()[ii*order_accuracy_ + jj] =
01251            phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01252          phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01253        }
01254      }
01255
01256      // 6.4. Insert stencil.
01257
01258      auto mm = 0;
01259      for (auto jj = num_bndy_approxs_; jj < num_bndy_approxs_ +  1; jj++) {
01260        for (auto ii = 0; ii < order_accuracy_ + 1; ii++) {
01261          if (ii == 0) {
01262            phi.data()[jj] = 0.0;
01263          } else {
01264            phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior_[ii - 1];
01265          }
01266        }
01267        mm++;
01268      }
01269
01270      #if MTK_VERBOSE_LEVEL > 4
01271      std::cout << "phi =" << std::endl;
01272      std::cout << phi << std::endl;
01273      #endif
01274
01276
```

```
01277     mtk::Real *lamed{};  // Used to build big lambda.
01278
01279     try {
01280       lamed = new mtk::Real[num_bndy_approxs_ - 1];
01281     } catch (std::bad_alloc &memory_allocation_exception) {
01282       std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01283         std::endl;
01284       std::cerr << memory_allocation_exception.what() << std::endl;
01285     }
01286     memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approxs_ - 1));
01287
01288     for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01289       lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01290     }
01291
01292     #if MTK_VERBOSE_LEVEL > 3
01293     std::cout << "lamed =" << std::endl;
01294     for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01295       std::cout << std::setw(12) << lamed[ii] << std::endl;
01296     }
01297     std::cout << std::endl;
01298     #endif
01299
01300     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01301       mtk::Real temp = mtk::kZero;
01302       for(auto jj = 0; jj < num_bndy_approxs_ - 1; ++jj) {
01303         temp = temp +
01304           lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01305       }
01306       hh[ii] = hh[ii] - temp;
01307     }
01308
01309     #if MTK_VERBOSE_LEVEL > 3
01310     std::cout << "big_lambda =" << std::endl;
01311     for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01312       std::cout << std::setw(12) << hh[ii] << std::endl;
01313     }
01314     std::cout << std::endl;
01315     #endif
01316
01318     int copy_result{};  // Should I replace the solution... not for now.
01319
01320
01321     mtk::Real normerr_; // Norm of the error for the solution on each row.
01322
01323     for(auto row_= 0; row_ < order_accuracy_ + 1; ++row_) {
01324       normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
    data(),
01325                                                           order_accuracy_ + 1,
01326                                                           order_accuracy_,
01327                                                           order_accuracy_,
01328                                                           hh,
01329                                                           weights_cbs_,
01330                                                           row_,
01331                                                           mimetic_threshold_,
01332                                                           copy_result);
01333       mtk::Real aux{normerr_/norm};
01334
01335       #if MTK_VERBOSE_LEVEL > 2
01336       std::cout << "Relative norm: " << aux << " " << std::endl;
01337       std::cout << std::endl;
01338       #endif
01339
01340       if (aux < minnorm) {
01341         minnorm = aux;
01342         minrow_= row_;
01343       }
01344     }
01345
01346     #if MTK_VERBOSE_LEVEL > 3
01347     std::cout << "weights_CBSA + lambda (after brute force search):" <<
01348       std::endl;
01349     for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01350       std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01351     }
01352     std::cout << std::endl;
01353     #endif
01354
01356
01357     // After we know which row yields the smallest relative norm that row is
01358     // chosen to be the objective function and the result of the optimizer is
```

```
01359      // chosen to be the new weights_.
01360
01361      #if MTK_VERBOSE_LEVEL > 2
01362      std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01363        minrow_ + 1 << std::endl;
01364      std::cout << std::endl;
01365      #endif
01366
01367      copy_result = 1;
01368      normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
   data(),
01369                                                          order_accuracy_ + 1,
01370                                                          order_accuracy_,
01371                                                          order_accuracy_,
01372                                                          hh,
01373                                                          weights_cbs_,
01374                                                          minrow_,
01375                                                          mimetic_threshold_,
01376                                                          copy_result);
01377      mtk::Real aux_{normerr_/norm};
01378      #if MTK_VERBOSE_LEVEL > 2
01379      std::cout << "Relative norm: " << aux_ << std::endl;
01380      std::cout << std::endl;
01381      #endif
01382
01383      delete [] lamed;
01384      lamed = nullptr;
01385    }
01386
01387    delete [] hh;
01388    hh = nullptr;
01389
01390    return true;
01391  }
01392
01393  bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01394
01395    #if MTK_VERBOSE_LEVEL > 3
01396    std::cout << "weights_* + lambda =" << std::endl;
01397    for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01398      std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01399    }
01400    std::cout << std::endl;
01401    #endif
01402
01403
01404
01405    mtk::Real *lambda{}; // Collection of bottom values from weights_.
01406
01407    try {
01408      lambda = new mtk::Real[dim_null_];
01409    } catch (std::bad_alloc &memory_allocation_exception) {
01410      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01411        std::endl;
01412      std::cerr << memory_allocation_exception.what() << std::endl;
01413    }
01414    memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01415
01416    for (auto ii = 0; ii < dim_null_; ++ii) {
01417      lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01418    }
01419
01420    #if MTK_VERBOSE_LEVEL > 3
01421    std::cout << "lambda =" << std::endl;
01422    for (auto ii = 0; ii < dim_null_; ++ii) {
01423      std::cout << std::setw(12) << lambda[ii] << std::endl;
01424    }
01425    std::cout << std::endl;
01426    #endif
01427
01428
01429    mtk::Real *alpha{}; // Collection of alpha values.
01430
01431    try {
01432      alpha = new mtk::Real[dim_null_];
01433    } catch (std::bad_alloc &memory_allocation_exception) {
01434      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01435        std::endl;
01436      std::cerr << memory_allocation_exception.what() << std::endl;
01437    }
01438    memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01439
01440
```

```
01441    for (auto ii = 0; ii < dim_null_; ++ii) {
01442      alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01443    }
01444
01445    #if MTK_VERBOSE_LEVEL > 3
01446    std::cout << "alpha =" << std::endl;
01447    for (auto ii = 0; ii < dim_null_; ++ii) {
01448      std::cout << std::setw(12) << alpha[ii] << std::endl;
01449    }
01450    std::cout << std::endl;
01451    #endif
01452
01454
01455    try {
01456      mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01457    } catch (std::bad_alloc &memory_allocation_exception) {
01458      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01459        std::endl;
01460      std::cerr << memory_allocation_exception.what() << std::endl;
01461    }
01462    memset(mim_bndy_,
01463           mtk::kZero,
01464           sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01465
01466    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01467      for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {
01468        mim_bndy_[ii*num_bndy_approxs_ + jj] =
01469          prem_apps_[ii*num_bndy_approxs_ + jj] +
01470          alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01471      }
01472    }
01473
01474    for(auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01475      mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01476        prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01477    }
01478
01479    #if MTK_VERBOSE_LEVEL > 4
01480    std::cout << "Collection of mimetic approximations:" << std::endl;
01481    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01482      for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01483        std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01484      }
01485      std::cout << std::endl;
01486    }
01487    std::cout << std::endl;
01488    #endif
01489
01490    delete[] lambda;
01491    lambda = nullptr;
01492
01493    delete[] alpha;
01494    alpha = nullptr;
01495
01496    return true;
01497 }
01498
01499 bool mtk::Grad1D::AssembleOperator(void) {
01500
01501    // The output array will have this form:
01502    // 1. The first entry of the array will contain the used order kk.
01503    // 2. The second entry of the array will contain the collection of
01504    // approximating coefficients for the interior of the grid.
01505    // 3. The third entry will contain a collection of weights.
01506    // 4. The next dim_null - 1 entries will contain the collections of
01507    // approximating coefficients for the west boundary of the grid.
01508
01509    gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01510      num_bndy_approxs_*num_bndy_coeffs_;
01511
01512    #if MTK_VERBOSE_LEVEL > 2
01513    std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01514    #endif
01515
01516    try {
01517      gradient_ = new mtk::Real[gradient_length_];
01518    } catch (std::bad_alloc &memory_allocation_exception) {
01519      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01520        std::endl;
01521      std::cerr << memory_allocation_exception.what() << std::endl;
01522    }
```

```
01523    memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01524
01526
01527    gradient_[0] = order_accuracy_;
01528
01531
01532    for (auto ii = 0; ii < order_accuracy_; ++ii) {
01533      gradient_[ii + 1] = coeffs_interior_[ii];
01534    }
01535
01537
01538    for (auto ii = 0; ii < order_accuracy_; ++ii) {
01539      gradient_[(order_accuracy_ + 1) + ii] = weights_cbs_[ii];
01540    }
01541
01544
01545    int offset{2*order_accuracy_ + 1};
01546
01547    int aux {}; // Auxiliary variable.
01548
01549    if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01550      for (auto ii = 0; ii < num_bndy_approxs_ ; ii++) {
01551        for (auto jj = 0; jj < num_bndy_coeffs_; jj++) {
01552          gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01553          aux++;
01554        }
01555      }
01556    } else {
01557      gradient_[offset + 0] = prem_apps_[0];
01558      gradient_[offset + 1] = prem_apps_[1];
01559      gradient_[offset + 2] = prem_apps_[2];
01560    }
01561
01562    #if MTK_VERBOSE_LEVEL > 1
01563    std::cout << "1D " << order_accuracy_ << "-order grad built!" << std::endl;
01564    std::cout << std::endl;
01565    #endif
01566
01567    return true;
01568 }
```

17.87   src/mtk_grad_2d.cc File Reference

Implements the class Grad2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_grad_1d.h"
#include "mtk_grad_2d.h"
```

Include dependency graph for mtk_grad_2d.cc:
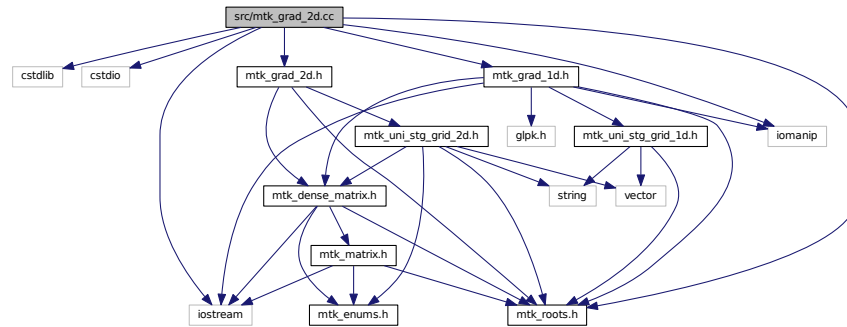
17.87.1 Detailed Description

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

Author

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

Definition in file mtk_grad_2d.cc.

17.88 mtk_grad_2d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
```

```
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_2d.h"
00066
00067 mtk::Grad2D::Grad2D():
00068   order_accuracy_(),
00069   mimetic_threshold_() {}
00070
00071 mtk::Grad2D::Grad2D(const Grad2D &grad):
00072   order_accuracy_(grad.order_accuracy_),
00073   mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad2D::~Grad2D() {}
00076
00077 bool mtk::Grad2D::ConstructGrad2D(const
     mtk::UniStgGrid2D &grid,
00078                                   int order_accuracy,
00079                                   mtk::Real mimetic_threshold) {
00080
00081   int num_cells_x = grid.num_cells_x();
00082   int num_cells_y = grid.num_cells_y();
00083
00084   int mx = num_cells_x + 1;  // Gx vertical dimension
00085   int nx = num_cells_x + 2;  // Gx horizontal dimension
00086   int my = num_cells_y + 1;  // Gy vertical dimension
00087   int ny = num_cells_y + 2;  // Gy horizontal dimension
00088
00089   mtk::Grad1D grad;
00090
00091   bool info = grad.ConstructGrad1D(order_accuracy, mimetic_threshold);
00092
00093   #ifdef MTK_PERFORM_PREVENTIONS
00094   if (!info) {
00095     std::cerr << "Mimetic grad could not be built." << std::endl;
00096     return info;
00097   }
00098   #endif
00099
00100   auto west = grid.west_bndy();
00101   auto east = grid.east_bndy();
00102   auto south = grid.south_bndy();
00103   auto north = grid.east_bndy();
00104
00105   mtk::UniStgGrid1D grid_x(west, east, num_cells_x);
00106   mtk::UniStgGrid1D grid_y(south, north, num_cells_y);
00107
00108   mtk::DenseMatrix Gx(grad.ReturnAsDenseMatrix(grid_x));
00109   mtk::DenseMatrix Gy(grad.ReturnAsDenseMatrix(grid_y));
00110
00111   bool padded{true};
00112   bool transpose{true};
00113
00114   mtk::DenseMatrix tix(num_cells_x, padded, transpose);
00115   mtk::DenseMatrix tiy(num_cells_y, padded, transpose);
00116
00117   mtk::DenseMatrix gxy(mtk::DenseMatrix::Kron(tiy, Gx));
00118   mtk::DenseMatrix gyx(mtk::DenseMatrix::Kron(Gy, tix));
00119
00120   #if MTK_VERBOSE_LEVEL > 2
00121   std::cout << "Gx: " << mx << " by " << nx << std::endl;
00122   std::cout << "Transpose Iy: " << num_cells_y << " by " << ny  << std::endl;
```

```
00123    std::cout << "Gy: " << my << " by " << ny << std::endl;
00124    std::cout << "Transpose Ix: " << num_cells_x << " by " << nx  << std::endl;
00125    std::cout << "Grad 2D: " << mx*num_cells_y + my*num_cells_x << " by " <<
00126      nx*ny <<std::endl;
00127    #endif
00128
00129    mtk::DenseMatrix g2d(mx*num_cells_y + my*num_cells_x, nx*ny);
00130
00131    for(auto ii = 0; ii < nx*ny; ii++) {
00132      for(auto jj = 0; jj < mx*num_cells_y; jj++) {
00133        g2d.SetValue(jj,ii, gxy.GetValue(jj,ii));
00134      }
00135      for(auto kk = 0; kk < my*num_cells_x; kk++) {
00136        g2d.SetValue(kk + mx*num_cells_y, ii, gyx.GetValue(kk,ii));
00137      }
00138    }
00139
00140    gradient_ = g2d;
00141
00142    return info;
00143 }
00144
00145 mtk::DenseMatrix mtk::Grad2D::ReturnAsDenseMatrix() const {
00146
00147    return gradient_;
00148 }
```

17.89   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:
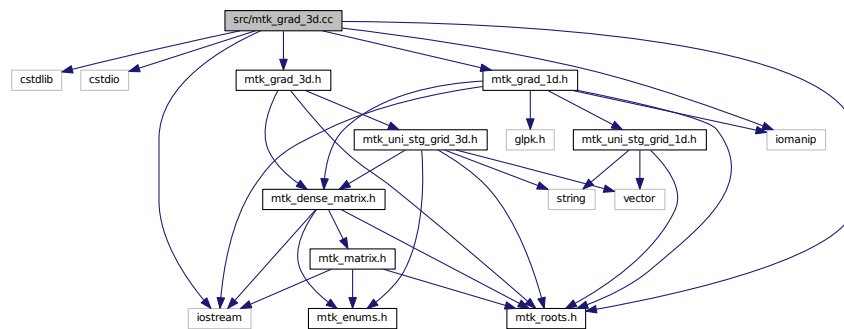
17.89.1   Detailed Description

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.

17.90 mtk_grad_3d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_grad_1d.h"
00065 #include "mtk_grad_3d.h"
00066
00067 mtk::Grad3D::Grad3D():
00068   order_accuracy_(),
00069   mimetic_threshold_() {}
00070
00071 mtk::Grad3D::Grad3D(const Grad3D &grad):
00072   order_accuracy_(grad.order_accuracy_),
00073   mimetic_threshold_(grad.mimetic_threshold_) {}
00074
00075 mtk::Grad3D::~Grad3D() {}
00076
00077 bool mtk::Grad3D::ConstructGrad3D(const
```

```
      mtk::UniStgGrid3D &grid,
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;  // 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    mtk::DenseMatrix aux1(mtk::DenseMatrix::Kron(tiz, tiy));
00127    mtk::DenseMatrix gx(mtk::DenseMatrix::Kron(aux1, Gx));
00128
00129
00131    mtk::DenseMatrix aux2(mtk::DenseMatrix::Kron(tiz, Gy));
00132    mtk::DenseMatrix gy(mtk::DenseMatrix::Kron(aux2, tix));
00133
00134
00136    mtk::DenseMatrix aux3(mtk::DenseMatrix::Kron(Gz, tiy));
00137    mtk::DenseMatrix gz(mtk::DenseMatrix::Kron(aux3, tix));
00138
00139
00140    #if MTK_VERBOSE_LEVEL > 2
00141    std::cout << "Gx: " << mx << " by " << nx << std::endl;
00142    std::cout << "Transpose Ix: " << num_cells_x << " by " << nx  << std::endl;
00143    std::cout << "Gy: " << my << " by " << ny << std::endl;
00144    std::cout << "Transpose Iy: " << num_cells_y << " by " << ny  << std::endl;
00145    std::cout << "Gz: " << mz << " by " << nz << std::endl;
00146    std::cout << "Transpose Iz: " << num_cells_z << " by " << nz  << std::endl;
00147    #endif
00148
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 }
```

17.91   src/mtk_interp_1d.cc File Reference

Includes the implementation of the class Interp1D.

```
#include <cstring>
#include "mtk_tools.h"
#include "mtk_interp_1d.h"
```
Include dependency graph for mtk_interp_1d.cc:
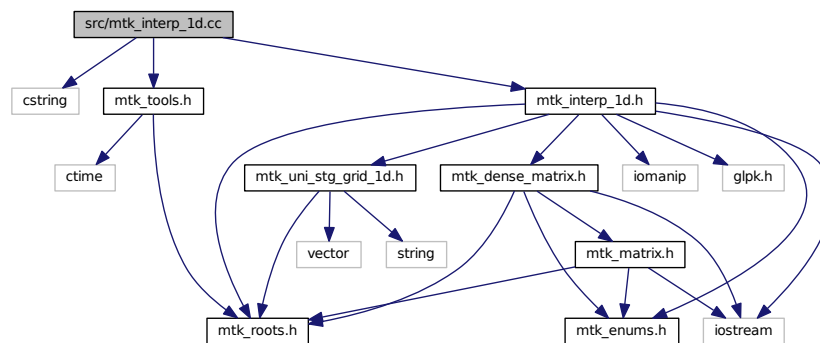
Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

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

17.91.1 Detailed Description

This class implements a 1D interpolation operator.

Author

> : Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu
> : Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file mtk_interp_1d.cc.

17.92 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
00069
00070   stream << "coeffs_interior_[1:" << in.order_accuracy_ << "] = ";
00071   for (auto ii = 0; ii < in.order_accuracy_; ++ii) {
00072     stream << std::setw(9) << in.coeffs_interior_[ii] << " ";
00073   }
00074   stream << std::endl;
00075
00076   return stream;
00077 }
00078 }
00079
00080 mtk::Interp1D::Interp1D():
00081   dir_interp_(mtk::SCALAR_TO_VECTOR),
00082   order_accuracy_(mtk::kDefaultOrderAccuracy),
00083   coeffs_interior_(nullptr) {}
00084
00085 mtk::Interp1D::Interp1D(const Interp1D &interp):
00086   dir_interp_(interp.dir_interp_),
00087   order_accuracy_(interp.order_accuracy_),
00088   coeffs_interior_(interp.coeffs_interior_) {}
00089
00090 mtk::Interp1D::~Interp1D() {
00091
00092   delete[] coeffs_interior_;
00093   coeffs_interior_ = nullptr;
00094 }
00095
00096 bool mtk::Interp1D::ConstructInterp1D(int order_accuracy,
00097   mtk::DirInterp dir) {
00097
00098   #if MTK_PERFORM_PREVENTIONS
00099   mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00100   mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00101   mtk::Tools::Prevent(dir < mtk::SCALAR_TO_VECTOR &&
00102                       dir > mtk::VECTOR_TO_SCALAR,
00103                       __FILE__, __LINE__, __func__);
00104   #endif
00105
00106   #if MTK_VERBOSE_LEVEL > 2
00107   std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00108   #endif
00109
00110   order_accuracy_ = order_accuracy;
00111
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::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 }
```

17.93 src/mtk_lap_1d.cc File Reference

Includes the implementation of the class Lap1D.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
```

Include dependency graph for mtk_lap_1d.cc:
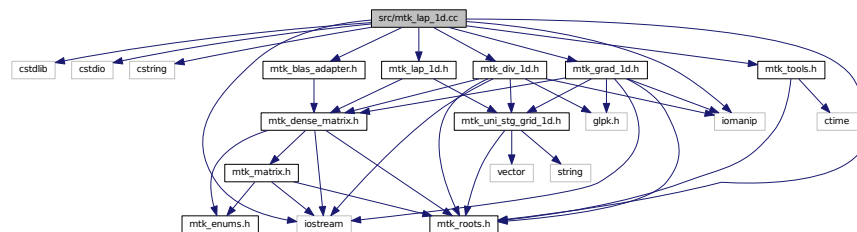
Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

- std::ostream & mtk::operator$<<$ (std::ostream &stream, mtk::Lap1D &in)

17.93.1 Detailed Description

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

Author

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

Definition in file mtk_lap_1d.cc.

17.94 mtk_lap_1d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

```
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059 #include <cstring>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_tools.h"
00066 #include "mtk_blas_adapter.h"
00067 #include "mtk_grad_1d.h"
00068 #include "mtk_div_1d.h"
00069 #include "mtk_lap_1d.h"
00070
00071 namespace mtk {
00072
00073 std::ostream& operator <<(std::ostream &stream, mtk::Lap1D &in) {
00074
00076
00077   stream << "laplacian_[0] = " << in.laplacian_[0] << std::endl << std::endl;
00078
00080
00081   stream << "laplacian_[1:" << 2*in.order_accuracy_ - 1 << "] = " <<
00082     std::endl << std::endl;
00083   for (auto ii = 1; ii <= (2*in.order_accuracy_ - 1); ++ii) {
00084     stream << std::setw(13) << in.laplacian_[ii] << " ";
00085   }
00086   stream << std::endl << std::endl;
00087
00089
00090   auto offset = 1 + (2*in.order_accuracy_ - 1);
00091
00092   stream << "laplacian_[" << offset << ":" << offset +
00093     (in.order_accuracy_ - 1)*(2*in.order_accuracy_) - 1 << "] = " <<
00094     std::endl << std::endl;
00095
00096   for (auto ii = 0; ii < in.order_accuracy_ - 1; ++ii) {
00097     for (auto jj = 0; jj < 2*in.order_accuracy_; ++jj) {
00098       stream << std::setw(13) <<
00099         in.laplacian_[offset + ii*(2*in.order_accuracy_) + jj];
00100     }
00101     stream << std::endl;
00102   }
00103
00104   return stream;
00105 }
00106 }
00107
00108 mtk::Lap1D::Lap1D():
00109   order_accuracy_(mtk::kDefaultOrderAccuracy),
00110   laplacian_length_(),
00111   delta_(mtk::kZero),
00112   mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00113
00114 mtk::Lap1D::~Lap1D() {
00115
00116   delete [] laplacian_;
00117   laplacian_ = nullptr;
00118 }
00119
00120 int mtk::Lap1D::order_accuracy() const {
00121
00122   return order_accuracy_;
00123 }
00124
00125 mtk::Real mtk::Lap1D::mimetic_threshold() const {
00126
00127   return mimetic_threshold_;
00128 }
00129
00130 mtk::Real mtk::Lap1D::delta() const {
00131
00132   return delta_;
00133 }
00134
00135 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00136                                 mtk::Real mimetic_threshold) {
```

```
00137
00138   #ifdef MTK_PERFORM_PREVENTIONS
00139   mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00140   mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00141   mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00142                       __FILE__, __LINE__, __func__);
00143
00144   if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00145     std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00146   }
00147
00148   std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00149   std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00150   #endif
00151
00152   order_accuracy_ = order_accuracy;
00153   mimetic_threshold_ = mimetic_threshold;
00154
00156   mtk::Grad1D grad; // Mimetic gradient.
00157
00158   bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00159
00160   #ifdef MTK_PERFORM_PREVENTIONS
00161   if (!info) {
00162     std::cerr << "Mimetic grad could not be built." << std::endl;
00163     return false;
00164   }
00165   #endif
00166
00168
00169   mtk::Div1D div; // Mimetic divergence.
00170
00171   info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00172
00173   #ifdef MTK_PERFORM_PREVENTIONS
00174   if (!info) {
00175     std::cerr << "Mimetic div could not be built." << std::endl;
00176     return false;
00177   }
00178   #endif
00179
00181
00182   // Since these are mimetic operator, we must multiply the matrices arising
00183   // from both the divergence and the Laplacian, in order to get the
00184   // approximating coefficients for the Laplacian operator.
00185
00186   // However, we must choose a grid that implied a step size of 1, so to get
00187   // the approximating coefficients, without being affected from the
00188   // normalization with respect to the grid (dimensionless).
00189
00190   // Also, the grid must be of the minimum size to support the requested order
00191   // of accuracy. We must please the divergence for this!
00192
00193   mtk::UniStgGrid1D aux(mtk::kZero,
00194                         (mtk::Real) 3*order_accuracy_ - 1,
00195                         3*order_accuracy_ - 1);
00196
00197   #if MTK_VERBOSE_LEVEL > 2
00198   std::cout << "aux =" << std::endl;
00199   std::cout << aux << std::endl;
00200   std::cout << "aux.delta_x() = " << aux.delta_x() << std::endl;
00201   std::cout << std::endl;
00202   #endif
00203
00204   mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00205
00206   #if MTK_VERBOSE_LEVEL > 4
00207   std::cout << "grad_m =" << std::endl;
00208   std::cout << grad_m << std::endl;
00209   #endif
00210
00211   mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux));
00212
00213   #if MTK_VERBOSE_LEVEL > 4
00214   std::cout << "div_m =" << std::endl;
00215   std::cout << div_m << std::endl;
00216   #endif
00217
00221
00222   mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00223
```

```
00224    lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00225
00226    #if MTK_VERBOSE_LEVEL > 4
00227    std::cout << "lap =" << std::endl;
00228    std::cout << lap << std::endl;
00229    #endif
00230
00232
00234
00235    // The output array will have this form:
00236    // 1. The first entry of the array will contain the used order kk.
00237    // 2. The second entry of the array will contain the collection of
00238    // approximating coefficients for the interior of the grid.
00239    // 3. The next entries will contain the collections of approximating
00240    // coefficients for the west boundary of the grid.
00241
00242    laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00243      (order_accuracy_ - 1)*(2*order_accuracy_);
00244
00245    #if MTK_VERBOSE_LEVEL > 2
00246    std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00247    std::cout << std::endl;
00248    #endif
00249
00250    try {
00251      laplacian_ = new mtk::Real[laplacian_length_];
00252    } catch (std::bad_alloc &memory_allocation_exception) {
00253      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00254        std::endl;
00255      std::cerr << memory_allocation_exception.what() << std::endl;
00256    }
00257    memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00258
00260
00261    laplacian_[0] = order_accuracy_;
00262
00265    for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00266
00267      laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00268    }
00269
00271
00272    auto offset = 1 + (2*order_accuracy_ - 1);
00273
00274    for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00275      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00276        laplacian_[offset + ii*(2*order_accuracy_) + jj] =
00277          lap.GetValue(1 + ii,jj);
00278      }
00279    }
00280
00281    delta_ = mtk::kZero;
00282
00283    return true;
00284 }
00285
00286 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(
00287    const UniStgGrid1D &grid) const {
00288
00289    int nn{grid.num_cells_x()};  // Number of cells on the grid.
00290
00291    #ifdef MTK_PERFORM_PREVENTIONS
00292    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00293    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00294    #endif
00295
00296    mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00297
00298    delta_ = grid.delta_x();
00299
00300    mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
      dx^2.
00301
00303
00304    auto offset = (1 + 2*order_accuracy_ - 1);
00305
00306    for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00307      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00308        lap.SetValue(1 + ii,
00309                     jj,
00310                     idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
```

```
00311     }
00312   }
00313
00315
00316   offset = 1 + (order_accuracy_ - 1);
00317
00318   int kk{1};
00319   for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00320     int mm{1};
00321     for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00322       lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00323       mm = mm + 1;
00324     }
00325     kk = kk + 1;
00326   }
00327
00329
00330   offset = (1 + 2*order_accuracy_ - 1);
00331
00332   auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00333
00334   auto ll = 1;
00335   auto rr = 1;
00336   for (auto ii = nn; ii > aux - 1; --ii) {
00337     auto cc = 0;
00338     for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00339       lap.SetValue(ii, jj, lap.GetValue(rr,cc));
00340       ++ll;
00341       ++cc;
00342     }
00343     rr++;
00344   }
00345
00352
00353   return lap;
00354 }
00355
00356 const mtk::Real* mtk::Lap1D::data(const UniStgGrid1D &grid) const {
00357
00358   mtk::DenseMatrix tmp;
00359
00360   tmp = ReturnAsDenseMatrix(grid);
00361
00362   return tmp.data();
00363 }
```

17.95   src/mtk_lap_2d.cc File Reference

Includes the implementation of the class Lap2D.

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <iomanip>
#include "mtk_roots.h"
#include "mtk_blas_adapter.h"
#include "mtk_grad_2d.h"
#include "mtk_div_2d.h"
#include "mtk_lap_2d.h"
```

Include dependency graph for mtk_lap_2d.cc:
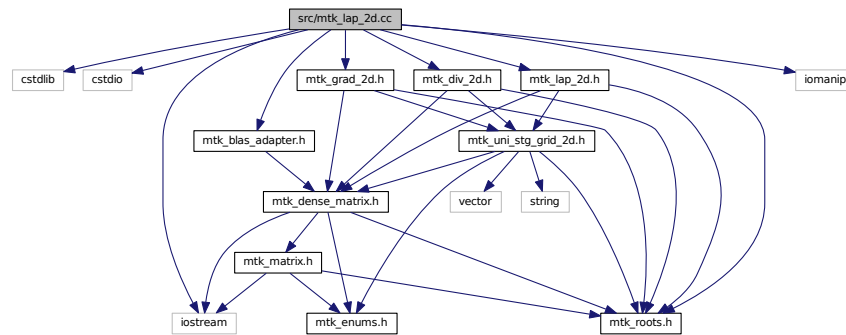
17.95.1 Detailed Description

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

Author

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

Definition in file mtk_lap_2d.cc.

17.96 mtk_lap_2d.cc

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed, unless these modifications are made through the project's GitHub
00021 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00022 should be developed and included in any deliverable.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 4. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders, and due credit should
00033 be given to the copyright holders.
00034
00035 5. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
```

```
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "mtk_roots.h"
00064 #include "mtk_blas_adapter.h"
00065 #include "mtk_grad_2d.h"
00066 #include "mtk_div_2d.h"
00067 #include "mtk_lap_2d.h"
00068
00069 mtk::Lap2D::Lap2D(): order_accuracy_(), mimetic_threshold_() {}
00070
00071 mtk::Lap2D::Lap2D(const Lap2D &lap):
00072   order_accuracy_(lap.order_accuracy_),
00073   mimetic_threshold_(lap.mimetic_threshold_) {}
00074
00075 mtk::Lap2D::~Lap2D() {}
00076
00077 bool mtk::Lap2D::ConstructLap2D(const
    mtk::UniStgGrid2D &grid,
00078                                  int order_accuracy,
00079                                  mtk::Real mimetic_threshold) {
00080
00081   mtk::Grad2D gg;
00082   mtk::Div2D dd;
00083
00084   bool info{gg.ConstructGrad2D(grid, order_accuracy, mimetic_threshold)};
00085
00086   #ifdef MTK_PERFORM_PREVENTIONS
00087   if (!info) {
00088     std::cerr << "Mimetic lap could not be built." << std::endl;
00089     return info;
00090   }
00091   #endif
00092
00093   info = dd.ConstructDiv2D(grid, order_accuracy, mimetic_threshold);
00094
00095   #ifdef MTK_PERFORM_PREVENTIONS
00096   if (!info) {
00097     std::cerr << "Mimetic div could not be built." << std::endl;
00098     return info;
00099   }
00100   #endif
00101
00102   mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00103   mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00104
00105   laplacian_ = mtk::BLASAdapter::RealDenseMM(ddm, ggm);
00106
00107   return info;
00108 }
00109
00110 mtk::DenseMatrix mtk::Lap2D::ReturnAsDenseMatrix() const {
00111
00112   return laplacian_;
00113 }
00114
00115 mtk::Real *mtk::Lap2D::data() const {
00116
00117   return laplacian_.data();
00118 }
```

17.97 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:
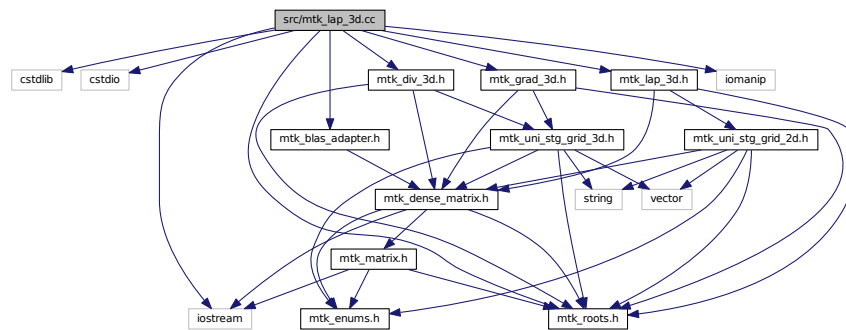
17.97.1 Detailed Description

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.

17.98 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  }
```

17.99 src/mtk_lapack_adapter.cc File Reference

Adapter class for the LAPACK API.

```
#include <cstring>
#include <iostream>
#include <iomanip>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_dense_matrix.h"
#include "mtk_lapack_adapter.h"
```

Include dependency graph for mtk_lapack_adapter.cc:

Namespaces

- **mtk**
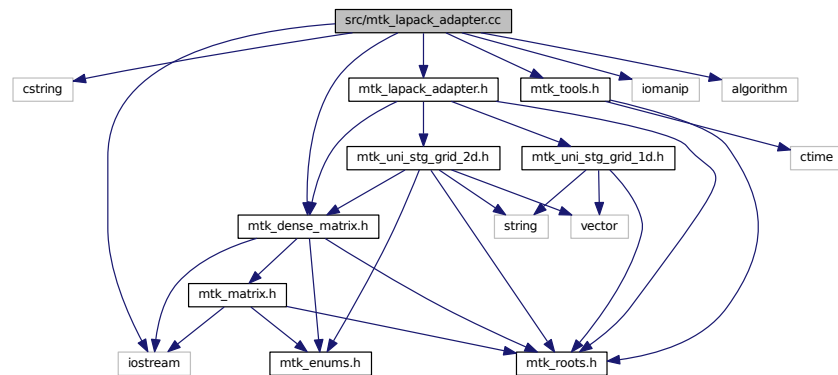
    Mimetic Methods Toolkit namespace.

Functions

- void mtk::sgesv_ (int ∗n, int ∗nrhs, Real ∗a, int ∗lda, int ∗ipiv, Real ∗b, int ∗ldb, int ∗info)
- void mtk::sgels_ (char ∗trans, int ∗m, int ∗n, int ∗nrhs, Real ∗a, int ∗lda, Real ∗b, int ∗ldb, Real ∗work, int ∗lwork, int ∗info)

    Single-precision GEneral matrix Least Squares solver.
- void mtk::sgeqrf_ (int ∗m, int ∗n, Real ∗a, int ∗lda, Real ∗tau, Real ∗work, int ∗lwork, int ∗info)

    Single-precision GEneral matrix QR Factorization.
- void mtk::sormqr_ (char ∗side, char ∗trans, int ∗m, int ∗n, int ∗k, Real ∗a, int ∗lda, Real ∗tau, Real ∗c, int ∗ldc, Real ∗work, int ∗lwork, int ∗info)

    Single-precision Orthogonal Matrix from QR factorization.

17.99.1 Detailed Description

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

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

See also

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

Todo Write documentation using LaTeX.

Author

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

Definition in file mtk_lapack_adapter.cc.

17.100 mtk_lapack_adapter.cc

```
00001
00021 /*
00022 Copyright (C) 2015, Computational Science Research Center, San Diego State
00023 University. All rights reserved.
00024
00025 Redistribution and use in source and binary forms, with or without modification,
00026 are permitted provided that the following conditions are met:
00027
00028 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00029 and a copy of the modified files should be reported once modifications are
00030 completed, unless these modifications are made through the project's GitHub
00031 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00032 should be developed and included in any deliverable.
00033
00034 2. Redistributions of source code must be done through direct
00035 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00036
00037 3. Redistributions in binary form must reproduce the above copyright notice,
00038 this list of conditions and the following disclaimer in the documentation and/or
00039 other materials provided with the distribution.
00040
00041 4. Usage of the binary form on proprietary applications shall require explicit
00042 prior written permission from the the copyright holders, and due credit should
00043 be given to the copyright holders.
```

```
00044
00045 5. Neither the name of the copyright holder nor the names of its contributors
00046 may be used to endorse or promote products derived from this software without
00047 specific prior written permission.
00048
00049 The copyright holders provide no reassurances that the source code provided does
00050 not infringe any patent, copyright, or any other intellectual property rights of
00051 third parties. The copyright holders disclaim any liability to any recipient for
00052 claims brought against recipient by any third party for infringement of that
00053 parties intellectual property rights.
00054
00055 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00056 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00057 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00058 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00059 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00060 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00061 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00062 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00063 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00064 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00065 */
00066
00067 #include <cstring>
00068
00069 #include <iostream>
00070 #include <iomanip>
00071
00072 #include <algorithm>
00073
00074 #include "mtk_tools.h"
00075 #include "mtk_dense_matrix.h"
00076 #include "mtk_lapack_adapter.h"
00077
00078 namespace mtk {
00079
00080 extern "C" {
00081
00082 #ifdef MTK_PRECISION_DOUBLE
00083
00102 void dgesv_(int* n,
00103             int* nrhs,
00104             Real* a,
00105             int* lda,
00106             int* ipiv,
00107             Real* b,
00108             int* ldb,
00109             int* info);
00110 #else
00111
00130 void sgesv_(int* n,
00131             int* nrhs,
00132             Real* a,
00133             int* lda,
00134             int* ipiv,
00135             Real* b,
00136             int* ldb,
00137             int* info);
00138 #endif
00139
00140 #ifdef MTK_PRECISION_DOUBLE
00141
00184 void dgels_(char* trans,
00185             int* m,
00186             int* n,
00187             int* nrhs,
00188             Real* a,
00189             int* lda,
00190             Real* b,
00191             int* ldb,
00192             Real* work,
00193             int* lwork,
00194             int* info);
00195 #else
00196
00239 void sgels_(char* trans,
00240             int* m,
00241             int* n,
00242             int* nrhs,
00243             Real* a,
00244             int* lda,
```

```
00245            Real* b,
00246            int* ldb,
00247            Real* work,
00248            int* lwork,
00249            int* info);
00250 #endif
00251
00252 #ifdef MTK_PRECISION_DOUBLE
00253
00282 void dgeqrf_(int *m,
00283            int *n,
00284            Real *a,
00285            int *lda,
00286            Real *tau,
00287            Real *work,
00288            int *lwork,
00289            int *info);
00290 #else
00291
00320 void sgeqrf_(int *m,
00321            int *n,
00322            Real *a,
00323            int *lda,
00324            Real *tau,
00325            Real *work,
00326            int *lwork,
00327            int *info);
00328 #endif
00329
00330 #ifdef MTK_PRECISION_DOUBLE
00331
00365 void dormqr_(char *side,
00366            char *trans,
00367            int *m,
00368            int *n,
00369            int *k,
00370            Real *a,
00371            int *lda,
00372            Real *tau,
00373            Real *c,
00374            int *ldc,
00375            Real *work,
00376            int *lwork,
00377            int *info);
00378 #else
00379
00413 void sormqr_(char *side,
00414            char *trans,
00415            int *m,
00416            int *n,
00417            int *k,
00418            Real *a,
00419            int *lda,
00420            Real *tau,
00421            Real *c,
00422            int *ldc,
00423            Real *work,
00424            int *lwork,
00425            int *info);
00426 #endif
00427 }
00428 }
00429
00430 int mtk::LAPACKAdapter::SolveDenseSystem(
     mtk::DenseMatrix &mm,
00431                                        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,
00466                                          mtk::DenseMatrix &bb) {
00467
00468   int nrhs{bb.num_rows()};  // Number of right-hand sides.
00469
00470   #ifdef MTK_PERFORM_PREVENTIONS
00471   mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00472   #endif
00473
00474   int *ipiv{};              // Array for pivoting information.
00475   int info{};              // Status of the solution.
00476   int mm_rank{mm.num_rows()}; // Rank of the matrix.
00477
00478   try {
00479     ipiv = new int[mm_rank];
00480   } catch (std::bad_alloc &memory_allocation_exception) {
00481     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00482       std::endl;
00483     std::cerr << memory_allocation_exception.what() << std::endl;
00484   }
00485   memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00486
00487   int ldbb = mm_rank;
00488   int mm_ld = mm_rank;
00489
00490   #ifdef MTK_PRECISION_DOUBLE
00491   dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00492   #else
00493   fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00494   #endif
00495
00496   delete [] ipiv;
00497
00498   // After output, the data in the matrix will be column-major ordered.
00499
00500   bb.SetOrdering(mtk::COL_MAJOR);
00501
00502   #if MTK_VERBOSE_LEVEL > 12
00503   std::cout << "bb_col_maj_ord =" << std::endl;
00504   std::cout << bb << std::endl;
00505   #endif
00506
00507   bb.OrderRowMajor();
00508
00509   #if MTK_VERBOSE_LEVEL > 12
00510   std::cout << "bb_row_maj_ord =" << std::endl;
00511   std::cout << bb << std::endl;
00512   #endif
00513
00514   return info;
00515 }
00516
00517 int mtk::LAPACKAdapter::SolveDenseSystem(mtk::DenseMatrix &mm,
00518                                          mtk::UniStgGrid1D &rhs) {
00519
00520   int nrhs{1};  // Number of right-hand sides.
00521
00522   int *ipiv{};              // Array for pivoting information.
00523   int info{};              // Status of the solution.
00524   int mm_rank{mm.num_rows()}; // Rank of the matrix.
00525
```

```
00526   try {
00527     ipiv = new int[mm_rank];
00528   } catch (std::bad_alloc &memory_allocation_exception) {
00529     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00530       std::endl;
00531     std::cerr << memory_allocation_exception.what() << std::endl;
00532   }
00533   memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00534
00535   int ldbb = mm_rank;
00536   int mm_ld = mm_rank;
00537
00538   mm.OrderColMajor();
00539
00540   #ifdef MTK_PRECISION_DOUBLE
00541   dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00542          rhs.discrete_field(), &ldbb, &info);
00543   #else
00544   fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00545          rhs.discrete_field(), &ldbb, &info);
00546   #endif
00547
00548   mm.OrderRowMajor();
00549
00550   delete [] ipiv;
00551
00552   return info;
00553 }
00554
00555 int mtk::LAPACKAdapter::SolveDenseSystem(
      mtk::DenseMatrix &mm,
00556                                         mtk::UniStgGrid2D &rhs) {
00557
00558   int nrhs{1};  // Number of right-hand sides.
00559
00560   int *ipiv{};                // Array for pivoting information.
00561   int info{};                 // Status of the solution.
00562   int mm_rank{mm.num_rows()}; // Rank of the matrix.
00563
00564   try {
00565     ipiv = new int[mm_rank];
00566   } catch (std::bad_alloc &memory_allocation_exception) {
00567     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00568       std::endl;
00569     std::cerr << memory_allocation_exception.what() << std::endl;
00570   }
00571   memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00572
00573   int ldbb = mm_rank;
00574   int mm_ld = mm_rank;
00575
00576   mm.OrderColMajor();
00577
00578   #ifdef MTK_PRECISION_DOUBLE
00579   dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00580          rhs.discrete_field(), &ldbb, &info);
00581   #else
00582   fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv,
00583          rhs.discrete_field(), &ldbb, &info);
00584   #endif
00585
00586   mm.OrderRowMajor();
00587
00588   delete [] ipiv;
00589
00590   return info;
00591 }
00592
00593 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
      (mtk::DenseMatrix &aa) {
00594
00595   mtk::Real *work{}; // Working array.
00596   mtk::Real *tau{};  // Array for the Householder scalars.
00597
00598   // Prepare to factorize: allocate and inquire for the value of lwork.
00599   try {
00600     work = new mtk::Real[1];
00601   } catch (std::bad_alloc &memory_allocation_exception) {
00602     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00603       std::endl;
00604     std::cerr << memory_allocation_exception.what() << std::endl;
```

```
00605    }
00606    memset(work, mtk::kZero, sizeof(aa.data()[0])*1);
00607
00608    int lwork{-1};
00609    int info{};
00610
00611    int aa_num_cols = aa.num_cols();
00612    int aaT_num_rows = aa.num_cols();
00613    int aaT_num_cols = aa.num_rows();
00614
00615    #if MTK_VERBOSE_LEVEL > 12
00616    std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00617    std::cout << aa << std::endl;
00618    #endif
00619
00620    #ifdef MTK_PRECISION_DOUBLE
00621    dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00622            tau,
00623            work, &lwork, &info);
00624    #else
00625    fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00626            tau,
00627            work, &lwork, &info);
00628    #endif
00629
00630    if (info == 0) {
00631      lwork = (int) work[0];
00632    } else {
00633      std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00634        std::endl;
00635      std::cerr << "Exiting..." << std::endl;
00636    }
00637
00638    #if MTK_VERBOSE_LEVEL > 10
00639    std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00640      << std::endl;
00641    #endif
00642
00643    delete [] work;
00644    work = nullptr;
00645
00646    // Once we know lwork, we can actually invoke the factorization:
00647    try {
00648      work = new mtk::Real [lwork];
00649    } catch (std::bad_alloc &memory_allocation_exception) {
00650      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00651        std::endl;
00652      std::cerr << memory_allocation_exception.what() << std::endl;
00653    }
00654    memset(work, mtk::kZero, sizeof(work[0])*lwork);
00655
00656    int ltau = std::min(aaT_num_rows,aaT_num_cols);
00657
00658    try {
00659      tau = new mtk::Real [ltau];
00660    } catch (std::bad_alloc &memory_allocation_exception) {
00661      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00662        std::endl;
00663      std::cerr << memory_allocation_exception.what() << std::endl;
00664    }
00665    memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00666
00667    #ifdef MTK_PRECISION_DOUBLE
00668    dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00669            tau, work, &lwork, &info);
00670    #else
00671    fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00672            tau, work, &lwork, &info);
00673    #endif
00674
00675    #ifdef MTK_PERFORM_PREVENTIONS
00676    if (!info) {
00677      std::cout << "QR factorization completed!" << std::endl << std::endl;
00678    } else {
00679      std::cerr << "Error solving system! info = " << info << std::endl;
00680      std::cerr << "Exiting..." << std::endl;
00681    }
00682    #endif
00683
00684    #if MTK_VERBOSE_LEVEL > 12
00685    std::cout << "Input matrix AFTER QR factorization:" << std::endl;
```

```
00686    std::cout << aa << std::endl;
00687    #endif
00688
00689    // We now generate the real matrix Q with orthonormal columns. This has to
00690    // be done separately since the actual output of dgeqrf_ (AA_) represents
00691    // the orthogonal matrix Q as a product of min(aa_num_rows,aa_num_cols)
00692    // elementary Householder reflectors. Notice that we must re-inquire the new
00693    // value for lwork that is used.
00694
00695    bool padded{false};
00696
00697    bool transpose{false};
00698
00699    mtk::DenseMatrix QQ_(aa.num_cols(),padded,transpose);
00700
00701    #if MTK_VERBOSE_LEVEL > 12
00702    std::cout << "Initialized QQ_T: " << std::endl;
00703    std::cout << QQ_ << std::endl;
00704    #endif
00705
00706    // Assemble the QQ_ matrix:
00707    lwork = -1;
00708
00709    delete[] work;
00710    work = nullptr;
00711
00712    try {
00713      work = new mtk::Real[1];
00714    } catch (std::bad_alloc &memory_allocation_exception) {
00715      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00716        std::endl;
00717      std::cerr << memory_allocation_exception.what() <<
00718        std::endl;
00719    }
00720    memset(work, mtk::kZero, sizeof(work[0])*1);
00721
00722    char side_{'L'};
00723    char trans_{'N'};
00724
00725    int aux = QQ_.num_rows();
00726
00727    #ifdef MTK_PRECISION_DOUBLE
00728    dormqr_(&side_, &trans_,
00729            &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00730            QQ_.data(), &aux, work, &lwork, &info);
00731    #else
00732    formqr_(&side_, &trans_,
00733            &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00734            QQ_.data(), &aux, work, &lwork, &info);
00735    #endif
00736
00737    if (info == 0) {
00738      lwork = (int) work[0];
00739    } else {
00740      std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00741      std::cerr << "Exiting..." << std::endl;
00742    }
00743
00744    #if MTK_VERBOSE_LEVEL > 10
00745    std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00746      std::endl << std::endl;
00747    #endif
00748
00749    delete[] work;
00750    work = nullptr;
00751
00752    try {
00753      work = new mtk::Real[lwork];
00754    } catch (std::bad_alloc &memory_allocation_exception) {
00755      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00756        std::endl;
00757      std::cerr << memory_allocation_exception.what() << std::endl;
00758    }
00759    memset(work, mtk::kZero, sizeof(work[0])*lwork);
00760
00761    #ifdef MTK_PRECISION_DOUBLE
00762    dormqr_(&side_, &trans_,
00763            &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00764            QQ_.data(), &aux, work, &lwork, &info);
00765    #else
00766    formqr_(&side_, &trans_,
```

```
00767              &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00768              QQ_.data(), &aux, work, &lwork, &info);
00769    #endif
00770
00771    #ifdef MTK_PERFORM_PREVENTIONS
00772    if (!info) {
00773      std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00774    } else {
00775      std::cerr << "Something went wrong solving system! info = " << info <<
00776        std::endl;
00777      std::cerr << "Exiting..." << std::endl;
00778    }
00779    #endif
00780
00781    delete[] work;
00782    work = nullptr;
00783
00784    delete[] tau;
00785    tau = nullptr;
00786
00787    return QQ_;
00788 }
00789
00790 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
      mtk::DenseMatrix &aa,
00791                                                    mtk::Real *ob_,
00792                                                    int ob_ld_) {
00793
00794    // We first invoke the solver to query for the value of lwork. For this,
00795    // we must at least allocate enough space to allow access to WORK(1), or
00796    // work[0]:
00797
00798    // If LWORK = -1, then a workspace query is assumed; the routine only
00799    // calculates the optimal size of the WORK array, returns this value as
00800    // the first entry of the WORK array, and no error message related to
00801    // LWORK is issued by XERBLA.
00802
00803    mtk::Real *work{}; // Work array.
00804
00805    try {
00806      work = new mtk::Real[1];
00807    } catch (std::bad_alloc &memory_allocation_exception) {
00808      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00809        std::endl;
00810      std::cerr << memory_allocation_exception.what() << std::endl;
00811    }
00812    memset(work, mtk::kZero, sizeof(work[0])*1);
00813
00814    char trans_{'N'};
00815    int nrhs_{1};
00816    int info{0};
00817    int lwork{-1};
00818
00819    int AA_num_rows_  = aa.num_cols();
00820    int AA_num_cols_  = aa.num_rows();
00821    int AA_ld_ = std::max(1,aa.num_cols());
00822
00823    #ifdef MTK_PRECISION_DOUBLE
00824    dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00825          ob_, &ob_ld_,
00826          work, &lwork, &info);
00827    #else
00828    sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00829          ob_, &ob_ld_,
00830          work, &lwork, &info);
00831    #endif
00832
00833    if (info == 0) {
00834      lwork = (int) work[0];
00835    } else {
00836      std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00837        std::endl;
00838      std::cerr << "Exiting..." << std::endl;
00839      return info;
00840    }
00841
00842    #if MTK_VERBOSE_LEVEL > 10
00843    std::cout << "lwork = " << std::endl << std::setw(12)<< lwork <<
00844      std::endl << std::endl;
00845    #endif
00846
```

```
00847    // We then use lwork's new value to create the work array:
00848    delete[] work;
00849    work = nullptr;
00850
00851    try {
00852      work = new mtk::Real[lwork];
00853    } catch (std::bad_alloc &memory_allocation_exception) {
00854      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00855      std::cerr << memory_allocation_exception.what() << std::endl;
00856    }
00857    memset(work, 0.0, sizeof(work[0])*lwork);
00858
00859    // We now invoke the solver again:
00860    #ifdef MTK_PRECISION_DOUBLE
00861    dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00862            ob_, &ob_ld_,
00863            work, &lwork, &info);
00864    #else
00865    sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00866            ob_, &ob_ld_,
00867            work, &lwork, &info);
00868    #endif
00869
00870    delete [] work;
00871    work = nullptr;
00872
00873    return info;
00874 }
```

17.101   src/mtk_matrix.cc File Reference

Implementing the representation of a matrix in the MTK.

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <iostream>
#include "mtk_tools.h"
#include "mtk_matrix.h"
```
Include dependency graph for mtk_matrix.cc:
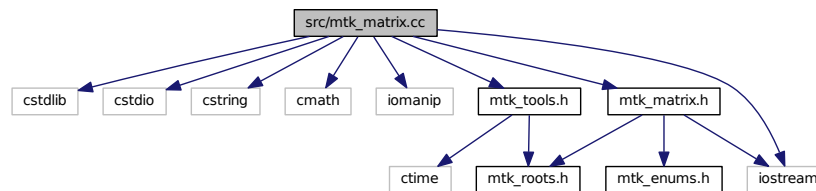
17.101.1   Detailed Description

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

Author

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

Definition in file mtk_matrix.cc.

17.102 mtk_matrix.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <cstdlib>
00057 #include <cstdio>
00058 #include <cstring>
00059 #include <cmath>
00060
00061 #include <iomanip>
00062 #include <iostream>
00063
00064 #include "mtk_tools.h"
00065 #include "mtk_matrix.h"
00066
00067 mtk::Matrix::Matrix():
00068   storage_(mtk::DENSE),
00069   ordering_(mtk::ROW_MAJOR),
00070   num_rows_(),
00071   num_cols_(),
00072   num_values_(),
00073   ld_(),
00074   num_zero_(),
00075   num_non_zero_(),
00076   num_null_(),
```

```
00077   num_non_null_(),
00078   kl_(),
00079   ku_(),
00080   bandwidth_(),
00081   abs_density_(),
00082   rel_density_(),
00083   abs_sparsity_(),
00084   rel_sparsity_() {}
00085
00086 mtk::Matrix::Matrix(const Matrix &in):
00087   storage_(in.storage_),
00088   ordering_(in.ordering_),
00089   num_rows_(in.num_rows_),
00090   num_cols_(in.num_cols_),
00091   num_values_(in.num_values_),
00092   ld_(in.ld_),
00093   num_zero_(in.num_zero_),
00094   num_non_zero_(in.num_non_zero_),
00095   num_null_(in.num_null_),
00096   num_non_null_(in.num_non_null_),
00097   kl_(in.kl_),
00098   ku_(in.ku_),
00099   bandwidth_(in.bandwidth_),
00100   abs_density_(in.abs_density_),
00101   rel_density_(in.rel_density_),
00102   abs_sparsity_(in.abs_sparsity_),
00103   rel_sparsity_(in.rel_sparsity_) {}
00104
00105 mtk::Matrix::~Matrix() 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::DENSE ||
00191                          ss == mtk::BANDED ||
00192                          ss == mtk::CRS),
00193                        __FILE__, __LINE__, __func__);
00194    #endif
00195
00196    storage_ = ss;
00197 }
00198
00199 void mtk::Matrix::set_ordering(const
      mtk::MatrixOrdering &oo) noexcept {
00200
00201    #ifdef MTK_PERFORM_PREVENTIONS
00202    mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
      mtk::COL_MAJOR),
00203                        __FILE__, __LINE__, __func__);
00204    #endif
00205
00206    ordering_ = oo;
00207
00208    ld_ = (ordering_ == mtk::ROW_MAJOR)?
00209      std::max(1,num_cols_): std::max(1,num_rows_);
00210 }
00211
00212 void mtk::Matrix::set_num_rows(const int &in) noexcept {
00213
00214    #ifdef MTK_PERFORM_PREVENTIONS
00215    mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00216    #endif
00217
00218    num_rows_ = in;
00219    num_values_ = num_rows_*num_cols_;
00220    ld_ = (ordering_ == mtk::ROW_MAJOR)?
00221      std::max(1,num_cols_): std::max(1,num_rows_);
00222 }
00223
00224 void mtk::Matrix::set_num_cols(const int &in) noexcept {
00225
00226    #ifdef MTK_PERFORM_PREVENTIONS
00227    mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00228    #endif
00229
00230    num_cols_ = in;
00231    num_values_ = num_rows_*num_cols_;
00232    ld_ = (ordering_ == mtk::ROW_MAJOR)?
00233      std::max(1,num_cols_): std::max(1,num_rows_);
00234 }
00235
```

```
00236 void mtk::Matrix::set_num_zero(const int &in) noexcept {
00237
00238   #ifdef MTK_PERFORM_PREVENTIONS
00239   mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00240   #endif
00241
00242   num_zero_ = in;
00243   num_non_zero_ = num_values_ - num_zero_;
00244
00246   rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00247   rel_sparsity_ = 1.0 - rel_density_;
00248 }
00249
00250 void mtk::Matrix::set_num_null(const int &in) noexcept {
00251
00252   #ifdef MTK_PERFORM_PREVENTIONS
00253   mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00254   #endif
00255
00256   num_null_ = in;
00257   num_non_null_ = num_values_ - num_null_;
00258
00260   abs_density_ = (mtk::Real) num_non_null_/num_values_;
00261   abs_sparsity_ = 1.0 - abs_density_;
00262 }
00263
00264 void mtk::Matrix::IncreaseNumZero() noexcept {
00265
00267
00268   num_zero_++;
00269   num_non_zero_ = num_values_ - num_zero_;
00270   rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00271   rel_sparsity_ = 1.0 - rel_density_;
00272 }
00273
00274 void mtk::Matrix::IncreaseNumNull() noexcept {
00275
00277
00278   num_null_++;
00279   num_non_null_ = num_values_ - num_null_;
00280   abs_density_ = (mtk::Real) num_non_null_/num_values_;
00281   abs_sparsity_ = 1.0 - abs_density_;
00282 }
```

17.103 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:

17.103.1 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.

17.104 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::Lap1D &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
00182   matrix.SetValue(0, 0, (west_coefficients_[0])(time));
00183
00185   matrix.SetValue(matrix.num_rows() - 1,
00186                   matrix.num_cols() - 1,
00187                   (east_coefficients_[0])(time));
00188
00190   if (highest_order_diff_west_ > 0) {
00191
00193     mtk::Grad1D grad;
00194     if (!grad.ConstructGrad1D(lap.order_accuracy(),
00195                               lap.mimetic_threshold())) {
00196       return false;
00197     }
00198
00200
00204     mtk::DenseMatrix coeffs(grad.mim_bndy());
00205
00206     mtk::Real idx = mtk::kOne/lap.delta();
00207
```

```
00209      for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00211        mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00214        mtk::Real unit_normal{-mtk::kOne};
00215        aux *= unit_normal*(west_coefficients_[1])(time);
00217        matrix.SetValue(0, ii, matrix.GetValue(0, ii) + aux);
00218      }
00219
00221
00226
00227      for (int ii = 0; ii < coeffs.num_cols(); ++ii) {
00229        mtk::Real aux{idx*coeffs.GetValue(0, ii)};
00233        mtk::Real unit_normal{mtk::kOne};
00234        aux *= -unit_normal*(east_coefficients_[1])(time);
00236        matrix.SetValue(matrix.num_rows() - 1,
00237                        matrix.num_rows() - 1 - ii,
00238                        matrix.GetValue(matrix.num_rows() - 1,
00239                                        matrix.num_rows() - 1 -ii) + aux);
00240      }
00241    }
00242
00243    return true;
00244 }
00245
00246 void mtk::RobinBCDescriptor1D::ImposeOnGrid(
00247      UniStgGrid1D &grid,
00248      const mtk::Real &time) const {
00249
00250    #ifdef MTK_PERFORM_PREVENTIONS
00251    mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00252    mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00253    mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00254    #endif
00255
00256    (grid.discrete_field())[0] = west_condition_(time);
00257    (grid.discrete_field())[grid.num_cells_x() + 1] = east_condition_(time);
00258 }
```

17.105   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:

17.105.1   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.

17.106 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()
    const noexcept {
00108
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(
00128     mtk::CoefficientFunction1D cw) {
00129
00130   #ifdef MTK_PERFORM_PREVENTIONS
00131   mtk::Tools::Prevent(cw == nullptr, __FILE__, __LINE__, __func__);
00132   mtk::Tools::Prevent(highest_order_diff_west_ > 1,
00133                       __FILE__, __LINE__, __func__);
00134   #endif
00135
00136   west_coefficients_.push_back(cw);
00137
00138   highest_order_diff_west_++;
00139 }
00140
00141 void mtk::RobinBCDescriptor2D::PushBackEastCoeff(
00142     mtk::CoefficientFunction1D ce) {
00143
00144   #ifdef MTK_PERFORM_PREVENTIONS
00145   mtk::Tools::Prevent(ce == nullptr, __FILE__, __LINE__, __func__);
00146   mtk::Tools::Prevent(highest_order_diff_east_ > 1,
00147                       __FILE__, __LINE__, __func__);
00148   #endif
00149
00150   east_coefficients_.push_back(ce);
00151
00152   highest_order_diff_east_++;
00153 }
00154
00155 void mtk::RobinBCDescriptor2D::PushBackSouthCoeff(
00156     mtk::CoefficientFunction1D cs) {
00157
00158   #ifdef MTK_PERFORM_PREVENTIONS
00159   mtk::Tools::Prevent(cs == nullptr, __FILE__, __LINE__, __func__);
00160   mtk::Tools::Prevent(highest_order_diff_south_ > 1,
00161                       __FILE__, __LINE__, __func__);
00162   #endif
00163
00164   south_coefficients_.push_back(cs);
00165
00166   highest_order_diff_south_++;
00167 }
00168
00169 void mtk::RobinBCDescriptor2D::PushBackNorthCoeff(
00170     mtk::CoefficientFunction1D cn) {
00171
00172   #ifdef MTK_PERFORM_PREVENTIONS
00173   mtk::Tools::Prevent(cn == nullptr, __FILE__, __LINE__, __func__);
00174   mtk::Tools::Prevent(highest_order_diff_north_ > 1,
00175                       __FILE__, __LINE__, __func__);
00176   #endif
00177
00178   north_coefficients_.push_back(cn);
00179
00180   highest_order_diff_north_++;
00181 }
00182
00183 void mtk::RobinBCDescriptor2D::set_west_condition(
00184     mtk::Real (*west_condition)(const mtk::Real &yy,
00185                                 const mtk::Real &tt)) noexcept {
00186
00187   #ifdef MTK_PERFORM_PREVENTIONS
```

```
00188    mtk::Tools::Prevent(west_condition == nullptr, __FILE__, __LINE__, __func__);
00189    #endif
00190
00191    west_condition_ = west_condition;
00192  }
00193
00194  void mtk::RobinBCDescriptor2D::set_east_condition(
00195      mtk::Real (*east_condition)(const mtk::Real &yy,
00196                                  const mtk::Real &tt)) noexcept {
00197
00198    #ifdef MTK_PERFORM_PREVENTIONS
00199    mtk::Tools::Prevent(east_condition == nullptr, __FILE__, __LINE__, __func__);
00200    #endif
00201
00202    east_condition_ = east_condition;
00203  }
00204
00205  void mtk::RobinBCDescriptor2D::set_south_condition(
00206      mtk::Real (*south_condition)(const mtk::Real &xx,
00207                                   const mtk::Real &tt)) noexcept {
00208
00209    #ifdef MTK_PERFORM_PREVENTIONS
00210    mtk::Tools::Prevent(south_condition == nullptr,
00211                        __FILE__, __LINE__, __func__);
00212    #endif
00213
00214    south_condition_ = south_condition;
00215  }
00216
00217  void mtk::RobinBCDescriptor2D::set_north_condition(
00218      mtk::Real (*north_condition)(const mtk::Real &xx,
00219                                   const mtk::Real &tt)) noexcept {
00220
00221    #ifdef MTK_PERFORM_PREVENTIONS
00222    mtk::Tools::Prevent(north_condition == nullptr,
00223                        __FILE__, __LINE__, __func__);
00224    #endif
00225
00226    north_condition_ = north_condition;
00227  }
00228
00229  bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryNoSpace
      (
00230      const mtk::Lap2D &lap,
00231      const mtk::UniStgGrid2D &grid,
00232      mtk::DenseMatrix &matrix,
00233      const mtk::Real &time) const {
00234
00236    // For the south-west corner:
00237    auto cc = (south_coefficients_[0])(grid.west_bndy(), time);
00238
00239
00240    #if MTK_VERBOSE_LEVEL > 2
00241    std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00242      matrix.num_cols() << " columns." << std::endl;
00243    std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00244    #endif
00245
00246    matrix.SetValue(0, 0, cc);
00247
00248    // Compute first centers per dimension.
00249    auto first_center_x = grid.west_bndy() + grid.delta_x()/
      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
00278    }
00280
00281    return true;
00282  }
00283
00284  bool mtk::RobinBCDescriptor2D::ImposeOnSouthBoundaryWithSpace
00      (
00285      const mtk::Lap2D &lap,
00286      const mtk::UniStgGrid2D &grid,
00287      mtk::DenseMatrix &matrix,
00288      const mtk::Real &time) const {
00289
00291
00294
00295    // For each entry on the diagonal:
00296    for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00297      // Evaluate next set spatial coordinates to evaluate the coefficient.
00298      mtk::Real xx{(grid.discrete_domain_x())[ii]};
00299      // Evaluate and assign the Dirichlet coefficient.
00300      mtk::Real cc = (south_coefficients_[0])(xx, time);
00301      matrix.SetValue(ii, ii, cc);
00302    }
00303
00304    if (highest_order_diff_south_ > 0) {
00305
00307    }
00308
00309    return true;
00310  }
00311
00312  bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryNoSpace
00      (
00313      const mtk::Lap2D &lap,
00314      const mtk::UniStgGrid2D &grid,
00315      mtk::DenseMatrix &matrix,
00316      const mtk::Real &time) const {
00317
00318    int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00319
00321
00322    // For the north-west corner:
00323    mtk::Real cc =
00324      (north_coefficients_[0])(grid.west_bndy(), time);
00325
00326    #if MTK_VERBOSE_LEVEL > 2
00327    std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00328      matrix.num_cols() << " columns." << std::endl;
00329    std::cout << "Setting at " << north_offset << ' ' << north_offset <<
00330      std::endl;
00331    #endif
00332
00333    matrix.SetValue(north_offset, north_offset, cc);
00334
00335    // Compute first centers per dimension.
00336    auto first_center_x = grid.west_bndy() + grid.delta_x()/
00      mtk::kTwo;
00337
00338    // For each entry on the diagonal (north boundary):
00339    for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00340      // Evaluate next set spatial coordinates to evaluate the coefficient.
00341      mtk::Real xx = first_center_x + ii*grid.delta_x();
00342      // Evaluate and assign the Dirichlet coefficient.
00343      cc = (north_coefficients_[0])(xx, time);
00344
00345      #if MTK_VERBOSE_LEVEL > 2
00346      std::cout << "Setting at " << north_offset + ii + 1 << ' ' <<
00347        north_offset + ii + 1 << std::endl;
00348      #endif
00349
00350      matrix.SetValue(north_offset + ii + 1, north_offset + ii + 1, cc);
00351    }
00352
```

```
00353    // For the north-east corner:
00354    cc = (north_coefficients_[0])(grid.east_bndy(), time);
00355
00356    #if MTK_VERBOSE_LEVEL > 2
00357    std::cout << "Setting at " << north_offset + grid.num_cells_x() + 1 <<
00358      ' ' << north_offset + grid.num_cells_x() + 1 << std::endl;
00359    #endif
00360
00361    matrix.SetValue(north_offset + grid.num_cells_x() + 1,
00362                    north_offset + grid.num_cells_x() + 1, cc);
00363
00364    if (highest_order_diff_north_ > 0) {
00365
00367    }
00368
00369    return true;
00370 }
00371
00372 bool mtk::RobinBCDescriptor2D::ImposeOnNorthBoundaryWithSpace
     (
00373      const mtk::Lap2D &lap,
00374      const mtk::UniStgGrid2D &grid,
00375      mtk::DenseMatrix &matrix,
00376      const mtk::Real &time) const {
00377
00379
00380    int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00381
00383    for (int ii = 0; ii < grid.num_cells_x() + 2; ++ii) {
00385      mtk::Real xx{(grid.discrete_domain_x())[ii]};
00387      mtk::Real cc = (north_coefficients_[0])(xx, time);
00388      matrix.SetValue(north_offset + ii, north_offset + ii, cc);
00389    }
00390
00391    if (highest_order_diff_north_ > 0) {
00392
00394    }
00395
00396    return true;
00397 }
00398
00399 bool mtk::RobinBCDescriptor2D::ImposeOnWestBoundaryNoSpace
     (
00400      const mtk::Lap2D &lap,
00401      const mtk::UniStgGrid2D &grid,
00402      mtk::DenseMatrix &matrix,
00403      const mtk::Real &time) const {
00404
00406
00407    // For the south-west corner:
00408    auto cc = (west_coefficients_[0])(grid.south_bndy(), time);
00409
00410    #if MTK_VERBOSE_LEVEL > 2
00411    std::cout << "Matrix has " << matrix.num_rows() << " rows and " <<
00412      matrix.num_cols() << " columns." << std::endl;
00413    std::cout << "Setting at " << 0 << ' ' << 0 << std::endl;
00414    #endif
00415
00419    mtk::Real harmonic_mean = mtk::kOne/matrix.GetValue(0, 0) +
     mtk::kOne/cc;
00420    harmonic_mean = mtk::kTwo/harmonic_mean;
00421
00423    matrix.SetValue(0, 0, harmonic_mean);
00424
00425    int west_offset{grid.num_cells_x() + 1};
00426
00427    auto first_center_y = grid.south_bndy() + grid.delta_y()/
     mtk::kTwo;
00428
00429    // For each west entry on the diagonal (west boundary):
00430    for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00431      // Evaluate next set spatial coordinates to evaluate the coefficient.
00432      mtk::Real yy = first_center_y + ii*grid.delta_y();
00433      // Evaluate and assign the Dirichlet coefficient.
00434      cc = (west_coefficients_[0])(yy, time);
00435
00436      #if MTK_VERBOSE_LEVEL > 2
00437      std::cout << "Setting at " << west_offset + ii + 1 << ' ' <<
00438        west_offset + ii + 1 << std::endl;
00439      #endif
```

```
00440
00441      matrix.SetValue(west_offset + ii + 1, west_offset + ii + 1, cc);
00442
00443      west_offset += grid.num_cells_x() + 1;
00444    }
00445
00446    // For the north-west corner:
00447    cc = (west_coefficients_[0])(grid.north_bndy(), time);
00448
00449    west_offset += grid.num_cells_x() + 1;
00450    int aux{west_offset};
00451    #if MTK_VERBOSE_LEVEL > 2
00452    std::cout << "Setting at " << aux << ' ' << aux << std::endl;
00453    #endif
00454
00455    harmonic_mean = mtk::kOne/matrix.GetValue(aux, aux) +
      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
00463    }
00464
00465    return true;
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
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
00490    }
00491
00492    return true;
00493 }
00494
00495 bool mtk::RobinBCDescriptor2D::ImposeOnEastBoundaryNoSpace
    (
00496      const mtk::Lap2D &lap,
00497      const mtk::UniStgGrid2D &grid,
00498      mtk::DenseMatrix &matrix,
00499      const mtk::Real &time) const {
00500
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
00559    }
00560
00561    return true;
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
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
00586    }
00587
00588    return true;
00589 }
00590
00591 bool mtk::RobinBCDescriptor2D::ImposeOnLaplacianMatrix(
00592      const mtk::Lap2D &lap,
00593      const mtk::UniStgGrid2D &grid,
00594      mtk::DenseMatrix &matrix,
00595      const mtk::Real &time) const {
00596
00597    #ifdef MTK_PERFORM_PREVENTIONS
00598    mtk::Tools::Prevent(highest_order_diff_south_ == -1,
00599                        __FILE__, __LINE__, __func__);
00600    mtk::Tools::Prevent(highest_order_diff_north_ == -1,
00601                        __FILE__, __LINE__, __func__);
00602    mtk::Tools::Prevent(highest_order_diff_west_ == -1,
```

```
00603                              __FILE__, __LINE__, __func__);
00604     mtk::Tools::Prevent(highest_order_diff_east_ == -1,
00605                              __FILE__, __LINE__, __func__);
00606     mtk::Tools::Prevent(grid.nature() != mtk::SCALAR,
00607                              __FILE__, __LINE__, __func__);
00608     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00609     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00610     mtk::Tools::Prevent(matrix.num_rows() == 0, __FILE__, __LINE__, __func__);
00611     mtk::Tools::Prevent(matrix.num_cols() == 0, __FILE__, __LINE__, __func__);
00612     #endif
00613
00616
00617     bool success{true};
00618
00619     if (!grid.Bound()) {
00620       success = ImposeOnSouthBoundaryNoSpace(lap, grid, matrix, time);
00621       #ifdef MTK_PERFORM_PREVENTIONS
00622       if (!success) {
00623         return false;
00624       }
00625       #endif
00626       success = ImposeOnNorthBoundaryNoSpace(lap, grid, matrix, time);
00627       #ifdef MTK_PERFORM_PREVENTIONS
00628       if (!success) {
00629         return false;
00630       }
00631       #endif
00632       success = ImposeOnWestBoundaryNoSpace(lap, grid, matrix, time);
00633       #ifdef MTK_PERFORM_PREVENTIONS
00634       if (!success) {
00635         return false;
00636       }
00637       #endif
00638       success = ImposeOnEastBoundaryNoSpace(lap, grid, matrix, time);
00639       #ifdef MTK_PERFORM_PREVENTIONS
00640       if (!success) {
00641         return false;
00642       }
00643       #endif
00644     } else {
00645       success = ImposeOnSouthBoundaryWithSpace(lap, grid, matrix, time);
00646       #ifdef MTK_PERFORM_PREVENTIONS
00647       if (!success) {
00648         return false;
00649       }
00650       #endif
00651       success = ImposeOnNorthBoundaryWithSpace(lap, grid, matrix, time);
00652       #ifdef MTK_PERFORM_PREVENTIONS
00653       if (!success) {
00654         return false;
00655       }
00656       #endif
00657       success = ImposeOnWestBoundaryWithSpace(lap, grid, matrix, time);
00658       #ifdef MTK_PERFORM_PREVENTIONS
00659       if (!success) {
00660         return false;
00661       }
00662       #endif
00663       success = ImposeOnEastBoundaryWithSpace(lap, grid, matrix, time);
00664       #ifdef MTK_PERFORM_PREVENTIONS
00665       if (!success) {
00666         return false;
00667       }
00668       #endif
00669     }
00670
00671     return success;
00672 }
00673
00674 void mtk::RobinBCDescriptor2D::ImposeOnGrid(
00675     mtk::UniStgGrid2D &grid,
00676     const mtk::Real &time) const {
00677
00678     #ifdef MTK_PERFORM_PREVENTIONS
00679     mtk::Tools::Prevent(grid.num_cells_x() == 0, __FILE__, __LINE__, __func__);
00680     mtk::Tools::Prevent(grid.num_cells_y() == 0, __FILE__, __LINE__, __func__);
00681     mtk::Tools::Prevent(west_condition_ == nullptr, __FILE__, __LINE__, __func__);
00682     mtk::Tools::Prevent(east_condition_ == nullptr, __FILE__, __LINE__, __func__);
00683     mtk::Tools::Prevent(south_condition_ == nullptr,
00684                              __FILE__, __LINE__, __func__);
00685     mtk::Tools::Prevent(north_condition_ == nullptr,
```

```
00686                        __FILE__, __LINE__, __func__);
00687    #endif
00688
00690    if (grid.nature() == mtk::SCALAR) {
00691
00693
00695      mtk::Real xx = grid.west_bndy();
00696      (grid.discrete_field())[0] = south_condition_(xx, time);
00697
00699      xx = xx + grid.delta_x()/mtk::kTwo;
00700      // For every point on the south boundary:
00701      for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00702        (grid.discrete_field())[ii + 1] =
00703          south_condition_(xx + ii*grid.delta_x(), time);
00704      }
00705
00707      xx = grid.east_bndy();
00708      (grid.discrete_field())[grid.num_cells_x() + 1] =
00709        south_condition_(xx, time);
00710
00712
00714      xx = grid.west_bndy();
00715      int north_offset{(grid.num_cells_y() + 1)*(grid.num_cells_x() + 2)};
00716      (grid.discrete_field())[north_offset] = north_condition_(xx, time);
00717
00719      xx = xx + grid.delta_x()/mtk::kTwo;
00720      for (int ii = 0; ii < grid.num_cells_x(); ++ii) {
00721        (grid.discrete_field())[north_offset + ii + 1] =
00722          north_condition_(xx + ii*grid.delta_x(), time);
00723      }
00724
00726      xx = grid.east_bndy();
00727      (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00728          north_condition_(xx, time);
00729
00731
00735      mtk::Real yy = grid.south_bndy();
00736      (grid.discrete_field())[0] =
00737        ((grid.discrete_field())[0] + west_condition_(yy, time))/
00738          mtk::kTwo;
00740      int west_offset{grid.num_cells_x() + 1 + 1};
00741      yy = yy + grid.delta_y()/mtk::kTwo;
00742      for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00743        #if MTK_VERBOSE_LEVEL > 2
00744        std::cout << "Adding on " << west_offset << "-th position." << std::endl;
00745        #endif
00746        (grid.discrete_field())[west_offset] =
00747          west_condition_(yy + ii*grid.delta_y(), time);
00748        west_offset += grid.num_cells_x() + 1 + 1;
00749      }
00750
00752      yy = grid.north_bndy();
00753      north_offset = (grid.num_cells_y() + 1)*(grid.num_cells_x() + 2);
00754      (grid.discrete_field())[north_offset] =
00755        ((grid.discrete_field())[north_offset] + west_condition_(yy, time))/
00756          mtk::kTwo;
00757
00759
00761      yy = grid.south_bndy();
00762      int east_offset{grid.num_cells_x() + 1};
00763      (grid.discrete_field())[east_offset] =
00764        ((grid.discrete_field())[east_offset] + east_condition_(yy, time))/
00765          mtk::kTwo;
00766
00768      yy = yy + grid.delta_y()/mtk::kTwo;
00769      for (int ii = 0; ii < grid.num_cells_y(); ++ii) {
00770        east_offset += grid.num_cells_x() + 1 + 1;
00771        #if MTK_VERBOSE_LEVEL > 2
00772        std::cout << "Adding on " << east_offset << "-th position." << std::endl;
00773        #endif
00774        (grid.discrete_field())[east_offset] =
00775          east_condition_(yy + ii*grid.delta_y(), time);
00776      }
00777
00779      yy = grid.north_bndy();
00780      (grid.discrete_field())[north_offset + grid.num_cells_x() + 1] =
00781        ((grid.discrete_field())[north_offset + grid.num_cells_x() + 1] +
00782        east_condition_(yy, time))/mtk::kTwo;
00783
00784    } else {
```

```
00785
00787
00789   }
00790 }
```

17.107   src/mtk_tools.cc File Reference

Implements a execution tool manager class.

```
#include <iostream>
#include "mtk_roots.h"
#include "mtk_tools.h"
```
Include dependency graph for mtk_tools.cc:
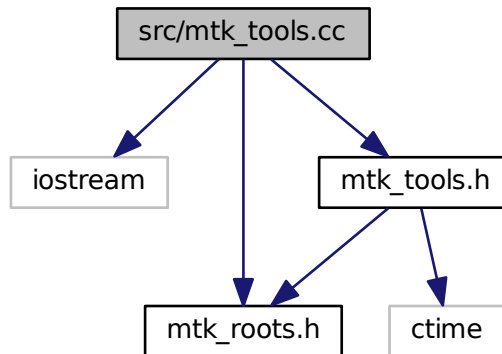
17.107.1   Detailed Description

Basic tools to ensure execution correctness.

Author

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

Definition in file mtk_tools.cc.

17.108   mtk_tools.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
```

```
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057
00058 #include "mtk_roots.h"
00059 #include "mtk_tools.h"
00060
00061 void mtk::Tools::Prevent(const bool condition,
00062                          const char *const fname,
00063                          int lineno,
00064                          const char *const fxname) noexcept {
00065
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
00081
00082 int mtk::Tools::test_number_;  // Used to control the correctness of the test.
00083
00084 mtk::Real mtk::Tools::duration_;  // Duration of the current test.
00085
00086 clock_t mtk::Tools::begin_time_;  // Used to time tests.
00087
00088 void mtk::Tools::BeginUnitTestNo(const int &nn) noexcept {
00089
00090   #if MTK_PERFORM_PREVENTIONS
00091   mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00092   #endif
00093
00094   test_number_ = nn;
00095
00096   std::cout << "Beginning test " << nn << "." << std::endl;
00097   begin_time_ = clock();
00098 }
00099
00100 void mtk::Tools::EndUnitTestNo(const int &nn) noexcept {
```

```
00101
00102   #if MTK_PERFORM_PREVENTIONS
00103   mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00104   #endif
00105
00106   duration_ = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00107 }
00108
00109 void mtk::Tools::Assert(const bool &condition) noexcept {
00110
00111   if (condition) {
00112     std::cout << "Test " << test_number_ << ": PASSED in " << duration_ <<
00113       " s." << std::endl;
00114   } else {
00115     std::cout << "Test " << test_number_ << ": FAILED in " << duration_ <<
00116       " s." << std::endl;
00117   }
00118 }
```

17.109 src/mtk_uni_stg_grid_1d.cc File Reference

Implementation of an 1D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include "mtk_roots.h"
#include "mtk_enums.h"
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_1d.h"
```
Include dependency graph for mtk_uni_stg_grid_1d.cc:

Namespaces

- mtk
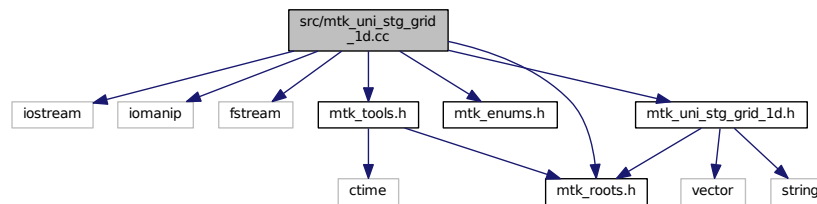
    Mimetic Methods Toolkit namespace.

Functions

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

17.109.1 Detailed Description

Implementation of an 1D uniform staggered grid.

Author

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

Definition in file mtk_uni_stg_grid_1d.cc.

17.110   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
00074
00075   stream << "x:";
00076   for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00077     stream << std::setw(10) << in.discrete_domain_x_[ii];
```

```
00078   }
00079   stream << std::endl;
00080
00082
00083   if (in.nature_ == mtk::SCALAR) {
00084     stream << "u:";
00085   }
00086   else {
00087     stream << "v:";
00088   }
00089   for (unsigned int ii = 0; ii < in.discrete_field_.size(); ++ii) {
00090     stream << std::setw(10) << in.discrete_field_[ii];
00091   }
00092
00093   stream << std::endl;
00094
00095   return stream;
00096 }
00097 }
00098
00099 mtk::UniStgGrid1D::UniStgGrid1D():
00100     nature_(),
00101     discrete_domain_x_(),
00102     discrete_field_(),
00103     west_bndy_x_(),
00104     east_bndy_x_(),
00105     num_cells_x_(),
00106     delta_x_() {}
00107
00108 mtk::UniStgGrid1D::UniStgGrid1D(const
    UniStgGrid1D &grid):
00109     nature_(grid.nature_),
00110     west_bndy_x_(grid.west_bndy_x_),
00111     east_bndy_x_(grid.east_bndy_x_),
00112     num_cells_x_(grid.num_cells_x_),
00113     delta_x_(grid.delta_x_) {
00114
00115     std::copy(grid.discrete_domain_x_.begin(),
00116             grid.discrete_domain_x_.begin() + grid.
    discrete_domain_x_.size(),
00117             discrete_domain_x_.begin());
00118
00119     std::copy(grid.discrete_field_.begin(),
00120             grid.discrete_field_.begin() + grid.discrete_field_.size(),
00121             discrete_field_.begin());
00122 }
00123
00124 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00125                               const Real &east_bndy_x,
00126                               const int &num_cells_x,
00127                               const mtk::FieldNature &nature) {
00128
00129   #ifdef MTK_PERFORM_PREVENTIONS
00130   mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00131   mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00132   mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00133   mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00134   #endif
00135
00136   nature_ = nature;
00137   west_bndy_x_ = west_bndy_x;
00138   east_bndy_x_ = east_bndy_x;
00139   num_cells_x_ = num_cells_x;
00140
00141   delta_x_ = (east_bndy_x - west_bndy_x)/((mtk::Real) num_cells_x);
00142 }
00143
00144 mtk::UniStgGrid1D::~UniStgGrid1D() {}
00145
00146 mtk::Real mtk::UniStgGrid1D::west_bndy_x() const {
00147
00148   return west_bndy_x_;
00149 }
00150
00151 mtk::Real mtk::UniStgGrid1D::east_bndy_x() const {
00152
00153   return east_bndy_x_;
00154 }
00155
00156 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00157
```

```
00158    return delta_x_;
00159 }
00160
00161 const mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() const
      {
00162
00163    return discrete_domain_x_.data();
00164 }
00165
00166 mtk::Real *mtk::UniStgGrid1D::discrete_field() {
00167
00168    return discrete_field_.data();
00169 }
00170
00171 int mtk::UniStgGrid1D::num_cells_x() const {
00172
00173    return num_cells_x_;
00174 }
00175
00176 void mtk::UniStgGrid1D::BindScalarField(
00177      mtk::Real (*ScalarField)(const mtk::Real &xx)) {
00178
00179    #ifdef MTK_PERFORM_PREVENTIONS
00180    mtk::Tools::Prevent(nature_ == mtk::VECTOR, __FILE__, __LINE__, __func__);
00181    #endif
00182
00184
00185    discrete_domain_x_.reserve(num_cells_x_ + 2);
00186
00187    discrete_domain_x_.push_back(west_bndy_x_);
00188    #ifdef MTK_PRECISION_DOUBLE
00189    auto first_center = west_bndy_x_ + delta_x_/2.0;
00190    #else
00191    auto first_center = west_bndy_x_ + delta_x_/2.0f;
00192    #endif
00193    discrete_domain_x_.push_back(first_center);
00194    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00195      discrete_domain_x_.push_back(first_center + ii*delta_x_);
00196    }
00197    discrete_domain_x_.push_back(east_bndy_x_);
00198
00200
00201    discrete_field_.reserve(num_cells_x_ + 2);
00202
00203    discrete_field_.push_back(ScalarField(west_bndy_x_));
00204
00205    discrete_field_.push_back(ScalarField(first_center));
00206    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00207      discrete_field_.push_back(ScalarField(first_center + ii*delta_x_));
00208    }
00209    discrete_field_.push_back(ScalarField(east_bndy_x_));
00210 }
00211
00212 void mtk::UniStgGrid1D::BindVectorField(
00213      mtk::Real (*VectorField)(mtk::Real xx)) {
00214
00215    #ifdef MTK_PERFORM_PREVENTIONS
00216    mtk::Tools::Prevent(nature_ == mtk::SCALAR, __FILE__, __LINE__, __func__);
00217    #endif
00218
00220
00221    discrete_domain_x_.reserve(num_cells_x_ + 1);
00222
00223    discrete_domain_x_.push_back(west_bndy_x_);
00224    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00225      discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00226    }
00227    discrete_domain_x_.push_back(east_bndy_x_);
00228
00230
00231    discrete_field_.reserve(num_cells_x_ + 1);
00232
00233    discrete_field_.push_back(VectorField(west_bndy_x_));
00234    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00235      discrete_field_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00236    }
00237    discrete_field_.push_back(VectorField(east_bndy_x_));
00238 }
00239
00240 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00241                                     std::string space_name,
```

```
00242                                      std::string field_name) const {
00243
00244   std::ofstream output_dat_file;  // Output file.
00245
00246   output_dat_file.open(filename);
00247
00248   if (!output_dat_file.is_open()) {
00249     return false;
00250   }
00251
00252   output_dat_file << "# " << space_name <<  ' ' << field_name << std::endl;
00253   for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00254     output_dat_file << discrete_domain_x_[ii] << ' ' << discrete_field_[ii] <<
00255       std::endl;
00256   }
00257
00258   output_dat_file.close();
00259
00260   return true;
00261 }
```

17.111 src/mtk_uni_stg_grid_2d.cc File Reference

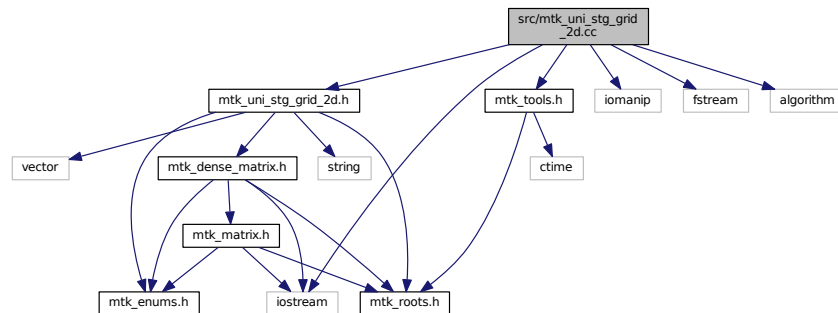Implementation of a 2D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_2d.h"
```
Include dependency graph for mtk_uni_stg_grid_2d.cc:

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

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

17.111.1 Detailed Description

Implementation of a 2D uniform staggered grid.

Author

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

Definition in file mtk_uni_stg_grid_2d.cc.

17.112 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
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
00090
00091    if (in.nature_ == mtk::SCALAR) {
00092      stream << "u:" << std::endl;
00093      if (in.discrete_field_.size() > 0) {
00094        for (int ii = 0; ii < in.num_cells_x_ + 2; ++ii) {
00095          for (int jj = 0; jj < in.num_cells_y_ + 2; ++jj) {
00096            stream << std::setw(10) << in.discrete_field_[ii*in.
    num_cells_y_ + jj];
00097          }
00098          stream << std::endl;
00099        }
00100      }
00101    } else {
00102
00103      int mm{in.num_cells_x_};
00104      int nn{in.num_cells_y_};
00105      int p_offset{nn*(mm + 1) - 1};
00106
00107      stream << "p(x,y):" << std::endl;
00108      for (int ii = 0; ii < nn; ++ii) {
00109        for (int jj = 0; jj < mm + 1; ++jj) {
00110          stream << std::setw(10) << in.discrete_field_[ii*(mm + 1) + jj];
00111        }
00112        stream << std::endl;
00113      }
00114      stream << std::endl;
00115
00116      stream << "q(x,y):" << std::endl;
00117      for (int ii = 0; ii < nn + 1; ++ii) {
00118        for (int jj = 0; jj < mm; ++jj) {
00119          stream << std::setw(10) <<
00120            in.discrete_field_[p_offset + ii*mm + jj];
00121        }
00122        stream << std::endl;
00123      }
00124      stream << std::endl;
00125    }
00126
00127    return stream;
00128 }
00129 }
00130
00131 mtk::UniStgGrid2D::UniStgGrid2D():
00132      discrete_domain_x_(),
00133      discrete_domain_y_(),
00134      discrete_field_(),
00135      nature_(),
00136      west_bndy_(),
00137      east_bndy_(),
00138      num_cells_x_(),
00139      delta_x_(),
00140      south_bndy_(),
00141      north_bndy_(),
00142      num_cells_y_(),
00143      delta_y_() {}
00144
00145 mtk::UniStgGrid2D::UniStgGrid2D(const
    UniStgGrid2D &grid):
00146      nature_(grid.nature_),
00147      west_bndy_(grid.west_bndy_),
00148      east_bndy_(grid.east_bndy_),
00149      num_cells_x_(grid.num_cells_x_),
00150      delta_x_(grid.delta_x_),
```

```
00151      south_bndy_(grid.south_bndy_),
00152      north_bndy_(grid.north_bndy_),
00153      num_cells_y_(grid.num_cells_y_),
00154      delta_y_(grid.delta_y_) {
00155
00156      std::copy(grid.discrete_domain_x_.begin(),
00157                grid.discrete_domain_x_.begin() + grid.
      discrete_domain_x_.size(),
00158                discrete_domain_x_.begin());
00159
00160      std::copy(grid.discrete_domain_y_.begin(),
00161                grid.discrete_domain_y_.begin() + grid.
      discrete_domain_y_.size(),
00162                discrete_domain_y_.begin());
00163
00164      std::copy(grid.discrete_field_.begin(),
00165                grid.discrete_field_.begin() + grid.discrete_field_.size(),
00166                discrete_field_.begin());
00167 }
00168
00169 mtk::UniStgGrid2D::UniStgGrid2D(const Real &west_bndy,
00170                                 const Real &east_bndy,
00171                                 const int &num_cells_x,
00172                                 const Real &south_bndy,
00173                                 const Real &north_bndy,
00174                                 const int &num_cells_y,
00175                                 const mtk::FieldNature &nature) {
00176
00177   #ifdef MTK_PERFORM_PREVENTIONS
00178   mtk::Tools::Prevent(west_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00179   mtk::Tools::Prevent(east_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00180   mtk::Tools::Prevent(east_bndy <= west_bndy, __FILE__, __LINE__, __func__);
00181   mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00182   mtk::Tools::Prevent(south_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00183   mtk::Tools::Prevent(north_bndy < mtk::kZero, __FILE__, __LINE__, __func__);
00184   mtk::Tools::Prevent(north_bndy <= south_bndy,
00185                       __FILE__, __LINE__, __func__);
00186   mtk::Tools::Prevent(num_cells_y < 0, __FILE__, __LINE__, __func__);
00187   #endif
00188
00189   nature_ = nature;
00190
00191   west_bndy_ = west_bndy;
00192   east_bndy_ = east_bndy;
00193   num_cells_x_ = num_cells_x;
00194
00195   south_bndy_ = south_bndy;
00196   north_bndy_ = north_bndy;
00197   num_cells_y_ = num_cells_y;
00198
00199   delta_x_ = (east_bndy_ - west_bndy_)/((mtk::Real) num_cells_x);
00200   delta_y_ = (north_bndy_ - south_bndy_)/((mtk::Real) num_cells_y);
00201 }
00202
00203 mtk::UniStgGrid2D::~UniStgGrid2D() {}
00204
00205 mtk::FieldNature mtk::UniStgGrid2D::nature() const {
00206
00207   return nature_;
00208 }
00209
00210 mtk::Real mtk::UniStgGrid2D::west_bndy() const {
00211
00212   return west_bndy_;
00213 }
00214
00215 mtk::Real mtk::UniStgGrid2D::east_bndy() const {
00216
00217   return east_bndy_;
00218 }
00219
00220 int mtk::UniStgGrid2D::num_cells_x() const {
00221
00222   return num_cells_x_;
00223 }
00224
00225 mtk::Real mtk::UniStgGrid2D::delta_x() const {
00226
00227   return delta_x_;
00228 }
00229
```

```
00230 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_x() const
      {
00231
00232   return discrete_domain_x_.data();
00233 }
00234
00235 mtk::Real mtk::UniStgGrid2D::south_bndy() const {
00236
00237   return south_bndy_;
00238 }
00239
00240 mtk::Real mtk::UniStgGrid2D::north_bndy() const {
00241
00242   return north_bndy_;
00243 }
00244
00245 int mtk::UniStgGrid2D::num_cells_y() const {
00246
00247   return num_cells_y_;
00248 }
00249
00250 mtk::Real mtk::UniStgGrid2D::delta_y() const {
00251
00252   return delta_y_;
00253 }
00254
00255 bool mtk::UniStgGrid2D::Bound() const {
00256
00257   return discrete_field_.size() != 0;
00258 }
00259
00260 const mtk::Real* mtk::UniStgGrid2D::discrete_domain_y() const
      {
00261
00262   return discrete_domain_y_.data();
00263 }
00264
00265 mtk::Real* mtk::UniStgGrid2D::discrete_field() {
00266
00267   return discrete_field_.data();
00268 }
00269
00270 int mtk::UniStgGrid2D::Size() const {
00271
00272   return discrete_field_.size();
00273 }
00274
00275 void mtk::UniStgGrid2D::BindScalarField(
00276     Real (*ScalarField)(const Real &xx, const Real &yy)) {
00277
00278   #ifdef MTK_PERFORM_PREVENTIONS
00279   mtk::Tools::Prevent(nature_ != mtk::SCALAR, __FILE__, __LINE__, __func__);
00280   #endif
00281
00283   discrete_domain_x_.reserve(num_cells_x_ + 2);
00285
00286   discrete_domain_x_.push_back(west_bndy_);
00287   #ifdef MTK_PRECISION_DOUBLE
00288   auto first_center = west_bndy_ + delta_x_/2.0;
00289   #else
00290   auto first_center = west_bndy_ + delta_x_/2.0f;
00291   #endif
00292   discrete_domain_x_.push_back(first_center);
00293   for (auto ii = 1; ii < num_cells_x_; ++ii) {
00294     discrete_domain_x_.push_back(first_center + ii*delta_x_);
00295   }
00296   discrete_domain_x_.push_back(east_bndy_);
00297
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
00315
00316     discrete_field_.reserve((num_cells_x_ + 2)*(num_cells_y_ + 2));
00317
00318     for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00319       for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00320         #if MTK_VERBOSE_LEVEL > 6
00321         std::cout << "Pushing value for x = " << discrete_domain_x_[jj] <<
00322           " y = " << discrete_domain_y_[ii] << std::endl;
00323         #endif
00324         discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00325                                               discrete_domain_y_[ii]));
00326       }
00327     }
00328 }
00329
00330 void mtk::UniStgGrid2D::BindVectorFieldPComponent(
00331     mtk::Real (*VectorField)(const mtk::Real &xx, const
00332     mtk::Real &yy)) {
00332
00333     int mm{num_cells_x_};
00334     int nn{num_cells_y_};
00335
00336     int total{nn*(mm + 1) + mm*(nn + 1)};
00337
00338     #ifdef MTK_PRECISION_DOUBLE
00339     double half_delta_x{delta_x_/2.0};
00340     double half_delta_y{delta_y_/2.0};
00341     #else
00342     float half_delta_x{delta_x_/2.0f};
00343     float half_delta_y{delta_y_/2.0f};
00344     #endif
00345
00347
00348     // We need every data point of the discrete domain; i.e. we need all the
00349     // nodes and all the centers. There are mm centers for the x direction, and
00350     // nn centers for the y direction. Since there is one node per center, that
00351     // amounts to 2*mm. If we finally consider the final boundary node, it
00352     // amounts to a total of 2*mm + 1 for the x direction. Analogously, for the
00353     // y direction, this amounts to 2*nn + 1.
00354
00355     discrete_domain_x_.reserve(2*mm + 1);
00356
00357     discrete_domain_x_.push_back(west_bndy_);
00358     for (int ii = 1; ii < (2*mm + 1); ++ii) {
00359       discrete_domain_x_.push_back(west_bndy_ + ii*half_delta_x);
00360     }
00361
00363
00364     discrete_domain_y_.reserve(2*nn + 1);
00365
00366     discrete_domain_y_.push_back(south_bndy_);
00367     for (int ii = 1; ii < (2*nn + 1); ++ii) {
00368       discrete_domain_y_.push_back(south_bndy_ + ii*half_delta_y);
00369     }
00370
00372
00373     discrete_field_.reserve(total);
00374
00375     // For each y-center.
00376     for (int ii = 1; ii < 2*nn + 1; ii += 2) {
00377
00378       // Bind all of the x-nodes for this y-center.
00379       for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00380         discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00381                                               discrete_domain_y_[ii]));
00382
00383         #if MTK_VERBOSE_LEVEL > 6
00384         std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00385           discrete_domain_y_[ii] << " = " <<
00386           VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00387         #endif
00388       }
00389     }
00390     #if MTK_VERBOSE_LEVEL > 6
00391     std::cout << std::endl;
00392     #endif
00393 }
00394
```

```
00395 void mtk::UniStgGrid2D::BindVectorFieldQComponent(
00396   mtk::Real (*VectorField)(const mtk::Real &xx, const
      mtk::Real &yy)) {
00397
00398   int mm{num_cells_x_};
00399   int nn{num_cells_y_};
00400
00402
00403   // For each y-node.
00404   for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00405
00406     // Bind all of the x-center for this y-node.
00407     for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00408       discrete_field_.push_back(VectorField(discrete_domain_x_[jj],
00409                                             discrete_domain_y_[ii]));
00410
00411       #if MTK_VERBOSE_LEVEL > 6
00412       std::cout << "Binding v at x = " << discrete_domain_x_[jj] << " y = " <<
00413         discrete_domain_y_[ii] << " = " <<
00414         VectorField(discrete_domain_x_[jj],discrete_domain_y_[ii]) << std::endl;
00415       #endif
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::VECTOR, __FILE__, __LINE__, __func__);
00429   #endif
00430
00431   BindVectorFieldPComponent(VectorFieldPComponent);
00432   BindVectorFieldQComponent(VectorFieldQComponent);
00433 }
00434
00435 bool mtk::UniStgGrid2D::WriteToFile(std::string filename,
00436                                    std::string space_name_x,
00437                                    std::string space_name_y,
00438                                    std::string field_name) const {
00439
00440   std::ofstream output_dat_file;  // Output file.
00441
00442   output_dat_file.open(filename);
00443
00444   if (!output_dat_file.is_open()) {
00445     return false;
00446   }
00447
00448   if (nature_ == mtk::SCALAR) {
00449     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00450       field_name << std::endl;
00451
00452     int idx{};
00453     for (unsigned int ii = 0; ii < discrete_domain_y_.size(); ++ii) {
00454       for (unsigned int jj = 0; jj < discrete_domain_x_.size(); ++jj) {
00455         output_dat_file << discrete_domain_x_[jj] << ' ' <<
00456                            discrete_domain_y_[ii] << ' ' <<
00457                            discrete_field_[idx] <<
00458                           std::endl;
00459         idx++;
00460       }
00461       output_dat_file << std::endl;
00462     }
00463   } else {
00464     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00465       field_name << std::endl;
00466
00467     output_dat_file << "# Horizontal component:" << std::endl;
00468
00469     int mm{num_cells_x_};
00470     int nn{num_cells_y_};
00471
00473
00474     // For each y-center.
00475     int idx{};
00476     for (int ii = 1; ii < 2*nn + 1; ii += 2) {
```

```
00477        // Bind all of the x-nodes for this y-center.
00478        for (int jj = 0; jj < 2*mm + 1; jj += 2) {
00479
00480          output_dat_file << discrete_domain_x_[jj] << ' ' <<
00481            discrete_domain_y_[ii] << ' ' << discrete_field_[idx] << ' ' <<
00482            mtk::kZero << std::endl;
00483
00484          ++idx;
00485        }
00486      }
00487
00489      int p_offset{nn*(mm + 1) - 1};
00490      idx = 0;
00491      output_dat_file << "# Vertical component:" << std::endl;
00492      // For each y-node.
00493      for (int ii = 0; ii < 2*nn + 1; ii += 2) {
00494        // Bind all of the x-center for this y-node.
00495        for (int jj = 1; jj < 2*mm + 1; jj += 2) {
00496
00497          output_dat_file << discrete_domain_x_[jj] << ' ' <<
00498            discrete_domain_y_[ii] << ' ' << mtk::kZero << ' ' <<
00499            discrete_field_[p_offset + idx] << std::endl;
00500
00501          ++idx;
00502        }
00503      }
00504    }
00505
00506    output_dat_file.close();
00507
00508    return true;
00509 }
```

17.113 src/mtk_uni_stg_grid_3d.cc File Reference
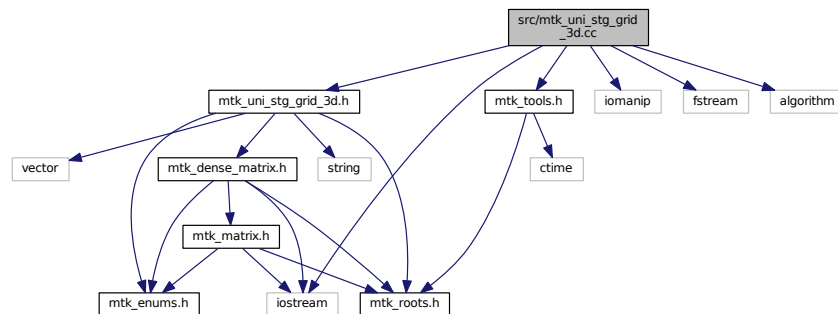
Implementation of a 2D uniform staggered grid.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include "mtk_tools.h"
#include "mtk_uni_stg_grid_3d.h"
```
Include dependency graph for mtk_uni_stg_grid_3d.cc:

Namespaces

- mtk

Mimetic Methods Toolkit namespace.

Functions

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

17.113.1 Detailed Description

Implementation of a 2D uniform staggered grid.

Author

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

Definition in file mtk_uni_stg_grid_3d.cc.

17.114 mtk_uni_stg_grid_3d.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
```

```
00059
00060 #include <algorithm>
00061
00062 #include "mtk_tools.h"
00063 #include "mtk_uni_stg_grid_3d.h"
00064
00065 namespace mtk {
00066
00067 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid3D &in) {
00068
00069   stream << '[' << in.west_bndy_ << ':' << in.num_cells_x_ << ':' <<
00070   in.east_bndy_ << "] x ";
00071
00072   stream << '[' << in.south_bndy_ << ':' << in.num_cells_y_ << ':' <<
00073   in.north_bndy_ << "] x ";
00074
00075   stream << '[' << in.bottom_bndy_ << ':' << in.num_cells_z_ << ':' <<
00076   in.top_bndy_ << "] = " << std::endl << std::endl;
00077
00079
00080   stream << "x:";
00081   for (auto const &cc: in.discrete_domain_x_) {
00082     stream << std::setw(10) << cc;
00083   }
00084   stream << std::endl;
00085
00086   stream << "y:";
00087   for (auto const &cc: in.discrete_domain_y_) {
00088     stream << std::setw(10) << cc;
00089   }
00090   stream << std::endl;
00091
00092   stream << "z:";
00093   for (auto const &cc: in.discrete_domain_z_) {
00094     stream << std::setw(10) << cc;
00095   }
00096   stream << std::endl;
00097
00099
00100   if (in.nature_ == mtk::SCALAR) {
00101     stream << "u(x,y,z):" << std::endl;
00102     if (in.discrete_field_.size() > 0) {
00103
00104     }
00105   } else {
00106     stream << "p(x,y,z):" << std::endl;
00107     stream << "q(x,y.z):" << std::endl;
00108     if (in.discrete_field_.size() > 0) {
00109
00110     }
00111   }
00112   return stream;
00113 }
00114 }
00115
00116 mtk::UniStgGrid3D mtk::UniStgGrid3D::operator=(const
      mtk::UniStgGrid3D &in) {
00117
00118   UniStgGrid3D out(in);
00119
00120   return out;
00121 }
00122
00123 mtk::UniStgGrid3D::UniStgGrid3D():
00124     discrete_domain_x_(),
00125     discrete_domain_y_(),
00126     discrete_domain_z_(),
00127     discrete_field_(),
00128     nature_(),
00129     west_bndy_(),
00130     east_bndy_(),
00131     num_cells_x_(),
00132     delta_x_(),
00133     south_bndy_(),
00134     north_bndy_(),
00135     num_cells_y_(),
00136     delta_y_(),
00137     bottom_bndy_(),
00138     top_bndy_(),
00139     num_cells_z_(),
00140     delta_z_() {}
```

```
00141
00142 mtk::UniStgGrid3D::UniStgGrid3D(const
      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::SCALAR, __FILE__, __LINE__, __func__);
00325   #endif
00326
00328
00329   discrete_domain_x_.reserve(num_cells_x_ + 2);
00330
00331   discrete_domain_x_.push_back(west_bndy_);
00332   #ifdef MTK_PRECISION_DOUBLE
00333   auto first_center = west_bndy_ + delta_x_/2.0;
00334   #else
00335   auto first_center = west_bndy_ + delta_x_/2.0f;
00336   #endif
00337   discrete_domain_x_.push_back(first_center);
00338   for (auto ii = 1; ii < num_cells_x_; ++ii) {
00339     discrete_domain_x_.push_back(first_center + ii*delta_x_);
00340   }
00341   discrete_domain_x_.push_back(east_bndy_);
00342
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
00360
00361   discrete_domain_z_.reserve(num_cells_z_ + 2);
00362
00363   discrete_domain_z_.push_back(bottom_bndy_);
00364   first_center = bottom_bndy_ + delta_z_/mtk::kTwo;
00365   discrete_domain_z_.push_back(first_center);
00366   for (auto ii = 1; ii < num_cells_z_; ++ii) {
00367     discrete_domain_z_.push_back(first_center + ii*delta_z_);
00368   }
00369   discrete_domain_z_.push_back(top_bndy_);
00370
00372
00373   int aux{(num_cells_x_ + 2)*(num_cells_y_ + 2)*(num_cells_z_ + 2)};
00374
00375   discrete_field_.reserve(aux);
00376
00377   for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00378     for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00379       for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00380         #if MTK_VERBOSE_LEVEL > 6
```

```
00381          std::cout << "At z = " << discrete_domain_z_[kk] << ": Pushing value"
00382            " for x = " << discrete_domain_x_[jj] << " y = " <<
00383          discrete_domain_y_[ii] << std::endl;
00384        #endif
00385        discrete_field_.push_back(ScalarField(discrete_domain_x_[jj],
00386                                              discrete_domain_y_[ii],
00387                                              discrete_domain_z_[kk]));
00388      }
00389    }
00390  }
00391 }
00392
00393 void mtk::UniStgGrid3D::BindVectorFieldPComponent(
00394   mtk::Real (*VectorField)(const mtk::Real &xx,
00395                            const mtk::Real &yy,
00396                            const mtk::Real &zz)) {
00397
00398 }
00399
00400 void mtk::UniStgGrid3D::BindVectorFieldQComponent(
00401   mtk::Real (*VectorField)(const mtk::Real &xx,
00402                            const mtk::Real &yy,
00403                            const mtk::Real &zz)) {
00404
00405 }
00406
00407 void mtk::UniStgGrid3D::BindVectorFieldRComponent(
00408   mtk::Real (*VectorField)(const mtk::Real &xx,
00409                            const mtk::Real &yy,
00410                            const mtk::Real &zz)) {
00411
00412 }
00413
00414 void mtk::UniStgGrid3D::BindVectorField(
00415   mtk::Real (*VectorFieldPComponent)(const mtk::Real &xx,
00416                                      const mtk::Real &yy,
00417                                      const mtk::Real &zz),
00418   mtk::Real (*VectorFieldQComponent)(const mtk::Real &xx,
00419                                      const mtk::Real &yy,
00420                                      const mtk::Real &zz),
00421   mtk::Real (*VectorFieldRComponent)(const mtk::Real &xx,
00422                                      const mtk::Real &yy,
00423                                      const mtk::Real &zz)) {
00424
00425   #ifdef MTK_PERFORM_PREVENTIONS
00426   mtk::Tools::Prevent(nature_ != mtk::VECTOR, __FILE__, __LINE__, __func__);
00427   #endif
00428
00429   BindVectorFieldPComponent(VectorFieldPComponent);
00430   BindVectorFieldQComponent(VectorFieldQComponent);
00431 }
00432
00433 bool mtk::UniStgGrid3D::WriteToFile(std::string filename,
00434                                    std::string space_name_x,
00435                                    std::string space_name_y,
00436                                    std::string space_name_z,
00437                                    std::string field_name) const {
00438
00439   std::ofstream output_dat_file;  // Output file.
00440
00441   output_dat_file.open(filename);
00442
00443   if (!output_dat_file.is_open()) {
00444     return false;
00445   }
00446
00447   if (nature_ == mtk::SCALAR) {
00448     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00449       space_name_z << ' ' << field_name << std::endl;
00450
00451   int idx{};
00452   for (int kk = 0; kk < num_cells_z_ + 2; ++kk) {
00453     for (int ii = 0; ii < num_cells_y_ + 2; ++ii) {
00454       for (int jj = 0; jj < num_cells_x_ + 2; ++jj) {
00455         output_dat_file << discrete_domain_x_[jj] << ' ' <<
00456           discrete_domain_y_[ii] << ' ' << discrete_domain_z_[kk] << ' ' <<
00457           discrete_field_[idx] << std::endl;
00458         idx++;
00459       }
00460     }
00461   }
```
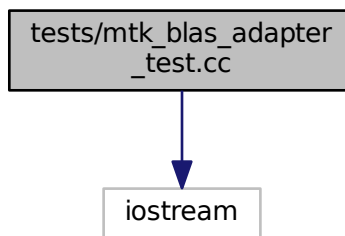
```
00462
00463   } else {
00464     output_dat_file << "# " << space_name_x << ' ' << space_name_y << ' ' <<
00465        space_name_z << ' ' << field_name << std::endl;
00466
00467   }
00468
00469   output_dat_file.close();
00470
00471   return true;
00472 }
```

17.115   tests/mtk_blas_adapter_test.cc File Reference

Test file for the mtk::BLASAdapter class.

```
#include <iostream>
```
Include dependency graph for mtk_blas_adapter_test.cc:

Functions

- int main ()

17.115.1   Detailed Description

Author

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

Definition in file mtk_blas_adapter_test.cc.

17.115.2   Function Documentation

17.115.2.1   int main (   )

Definition at line 109 of file mtk_blas_adapter_test.cc.

17.116 mtk_blas_adapter_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestRealDenseMM() {
00061
00062   mtk::Tools::BeginUnitTestNo(1);
00063
00064   int rr = 2;
00065   int cc = 3;
00066
00067   mtk::DenseMatrix aa(rr,cc);
00068
00069   aa.SetValue(0,0,1.0);
00070   aa.SetValue(0,1,2.0);
00071   aa.SetValue(0,2,3.0);
00072   aa.SetValue(1,0,4.0);
00073   aa.SetValue(1,1,5.0);
00074   aa.SetValue(1,2,6.0);
00075
00076   mtk::DenseMatrix bb(cc,rr);
00077
00078   bb.SetValue(0,0,7.0);
00079   bb.SetValue(0,1,8.0);
00080   bb.SetValue(1,0,9.0);
00081   bb.SetValue(1,1,10.0);
00082   bb.SetValue(2,0,11.0);
00083   bb.SetValue(2,1,12.0);
00084
```

```
00085    mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00086
00087    mtk::DenseMatrix ff(rr,rr);
00088
00089    ff.SetValue(0,0,58.0);
00090    ff.SetValue(0,1,64.00);
00091    ff.SetValue(1,0,139.0);
00092    ff.SetValue(1,1,154.0);
00093
00094    mtk::Tools::EndUnitTestNo(1);
00095    mtk::Tools::Assert(pp == ff);
00096 }
00097
00098 int main () {
00099
00100    std::cout << "Testing mtk::BLASAdapter class." << std::endl;
00101
00102    TestRealDenseMM();
00103 }
00104
00105 #else
00106 #include <iostream>
00107 using std::cout;
00108 using std::endl;
00109 int main () {
00110    cout << "This code HAS to be compiled with support for C++11." << endl;
00111    cout << "Exiting..." << endl;
00112 }
00113 #endif
```
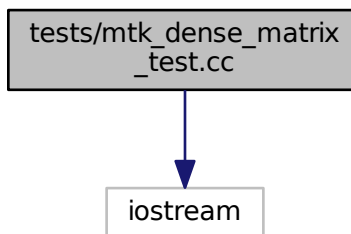
17.117 tests/mtk_dense_matrix_test.cc File Reference

Test file for the mtk::DenseMatrix class.

```
#include <iostream>
```
Include dependency graph for mtk_dense_matrix_test.cc:

Functions

- int main ()

17.117.1 Detailed Description

Author

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

Definition in file mtk_dense_matrix_test.cc.

17.117.2 Function Documentation

17.117.2.1 int main ( )

Definition at line 330 of file mtk_dense_matrix_test.cc.

17.118 mtk_dense_matrix_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062
00063   mtk::Tools::BeginUnitTestNo(1);
```

```
00064
00065   mtk::DenseMatrix m1;
00066
00067   mtk::Tools::EndUnitTestNo(1);
00068   mtk::Tools::Assert(m1.data() == nullptr);
00069 }
00070
00071 void TestConstructorWithNumRowsNumCols() {
00072
00073   mtk::Tools::BeginUnitTestNo(2);
00074
00075   int rr = 4;
00076   int cc = 7;
00077
00078   mtk::DenseMatrix m2(rr,cc);
00079
00080   mtk::Tools::EndUnitTestNo(2);
00081
00082   bool assertion =
00083     m2.data() != nullptr && m2.num_rows() == rr && m2.num_cols() == cc;
00084
00085   mtk::Tools::Assert(assertion);
00086 }
00087
00088 void TestConstructAsIdentity() {
00089
00090   mtk::Tools::BeginUnitTestNo(3);
00091
00092   int rank = 5;
00093   bool padded = true;
00094   bool transpose = false;
00095
00096   mtk::DenseMatrix m3(rank,padded,transpose);
00097
00098   mtk::DenseMatrix rr(rank + 2,rank);
00099
00100   for (int ii = 0; ii < rank; ++ii) {
00101     rr.SetValue(ii + 1, ii, mtk::kOne);
00102   }
00103
00104   mtk::Tools::EndUnitTestNo(3);
00105   mtk::Tools::Assert(m3 == rr);
00106 }
00107
00108   void TestConstructAsVandermonde() {
00109
00110   mtk::Tools::BeginUnitTestNo(4);
00111
00112   int rank = 5;
00113   bool padded = false;
00114   bool transpose = false;
00115
00116   mtk::DenseMatrix m4(rank,padded,transpose);
00117
00118   mtk::DenseMatrix rr(rank,rank);
00119
00120   for (int ii = 0; ii < rank; ++ii) {
00121     rr.SetValue(ii, ii, mtk::kOne);
00122   }
00123
00124   mtk::Tools::EndUnitTestNo(4);
00125   mtk::Tools::Assert(m4 == rr);
00126 }
00127
00128 void TestSetValueGetValue() {
00129
00130   mtk::Tools::BeginUnitTestNo(5);
00131
00132   int rr = 4;
00133   int cc = 7;
00134
00135   mtk::DenseMatrix m5(rr,cc);
00136
00137   for (auto ii = 0; ii < rr; ++ii) {
00138     for (auto jj = 0; jj < cc; ++jj) {
00139       m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00140     }
00141   }
00142
00143   mtk::Real *vals = m5.data();
00144
```

```
00145   bool assertion{true};
00146
00147   for (auto ii = 0; ii < rr && assertion; ++ii) {
00148     for (auto jj = 0; jj < cc  && assertion; ++jj) {
00149       assertion = assertion && m5.GetValue(ii,jj) == vals[ii*cc + jj];
00150     }
00151   }
00152
00153   mtk::Tools::EndUnitTestNo(5);
00154   mtk::Tools::Assert(assertion);
00155 }
00156
00157 void TestConstructAsVandermondeTranspose() {
00158
00159   mtk::Tools::BeginUnitTestNo(6);
00160
00161   bool transpose = false;
00162   int generator_length = 3;
00163   int progression_length = 4;
00164
00165   mtk::Real generator[] = {-0.5, 0.5, 1.5};
00166
00167   mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00168
00169   transpose = true;
00170
00171   mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00172   mtk::DenseMatrix rr(progression_length, generator_length);
00173
00174   rr.SetValue(0, 0, 1.0);
00175   rr.SetValue(0, 1, 1.0);
00176   rr.SetValue(0, 2, 1.0);
00177
00178   rr.SetValue(1, 0, -0.5);
00179   rr.SetValue(1, 1, 0.5);
00180   rr.SetValue(1, 2, 1.5);
00181
00182   rr.SetValue(2, 0, 0.25);
00183   rr.SetValue(2, 1, 0.25);
00184   rr.SetValue(2, 2, 2.25);
00185
00186   rr.SetValue(3, 0, -0.125);
00187   rr.SetValue(3, 1, 0.125);
00188   rr.SetValue(3, 2, 3.375);
00189
00190   mtk::Tools::EndUnitTestNo(6);
00191   mtk::Tools::Assert(m7 == rr);
00192 }
00193
00194 void TestKron() {
00195
00196   mtk::Tools::BeginUnitTestNo(7);
00197
00198   bool padded = false;
00199   bool transpose = false;
00200   int lots_of_rows = 2;
00201   int lots_of_cols = 5;
00202   mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00203
00204   mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00205
00206   for (auto ii = 0; ii < lots_of_rows; ++ii) {
00207     for (auto jj = 0; jj < lots_of_cols; ++jj) {
00208       m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00209     }
00210   }
00211
00212   mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00213
00214   mtk::DenseMatrix rr(lots_of_rows*lots_of_rows, lots_of_rows*lots_of_cols);
00215
00216   rr.SetValue(0,0,1.0);
00217   rr.SetValue(0,1,2.0);
00218   rr.SetValue(0,2,3.0);
00219   rr.SetValue(0,3,4.0);
00220   rr.SetValue(0,4,5.0);
00221   rr.SetValue(0,5,0.0);
00222   rr.SetValue(0,6,0.0);
00223   rr.SetValue(0,7,0.0);
00224   rr.SetValue(0,8,0.0);
00225   rr.SetValue(0,9,0.0);
```

```
00226
00227    rr.SetValue(1,0,6.0);
00228    rr.SetValue(1,1,7.0);
00229    rr.SetValue(1,2,8.0);
00230    rr.SetValue(1,3,9.0);
00231    rr.SetValue(1,4,10.0);
00232    rr.SetValue(1,5,0.0);
00233    rr.SetValue(1,6,0.0);
00234    rr.SetValue(1,7,0.0);
00235    rr.SetValue(1,8,0.0);
00236    rr.SetValue(1,9,0.0);
00237
00238    rr.SetValue(2,0,0.0);
00239    rr.SetValue(2,1,0.0);
00240    rr.SetValue(2,2,0.0);
00241    rr.SetValue(2,3,0.0);
00242    rr.SetValue(2,4,0.0);
00243    rr.SetValue(2,5,1.0);
00244    rr.SetValue(2,6,2.0);
00245    rr.SetValue(2,7,3.0);
00246    rr.SetValue(2,8,4.0);
00247    rr.SetValue(2,9,5.0);
00248
00249    rr.SetValue(3,0,0.0);
00250    rr.SetValue(3,1,0.0);
00251    rr.SetValue(3,2,0.0);
00252    rr.SetValue(3,3,0.0);
00253    rr.SetValue(3,4,0.0);
00254    rr.SetValue(3,5,6.0);
00255    rr.SetValue(3,6,7.0);
00256    rr.SetValue(3,7,8.0);
00257    rr.SetValue(3,8,9.0);
00258    rr.SetValue(3,9,10.0);
00259
00260    mtk::Tools::EndUnitTestNo(7);
00261    mtk::Tools::Assert(m10 == rr);
00262 }
00263
00264 void TestConstructWithNumRowsNumColsAssignmentOperator() {
00265
00266    mtk::Tools::BeginUnitTestNo(8);
00267
00268    int lots_of_rows = 4;
00269    int lots_of_cols = 3;
00270    mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00271
00272    for (auto ii = 0; ii < lots_of_rows; ++ii) {
00273      for (auto jj = 0; jj < lots_of_cols; ++jj) {
00274        m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00275      }
00276    }
00277
00278    m11.Transpose();
00279
00280    mtk::DenseMatrix m12;
00281
00282    m12 = m11;
00283
00284    mtk::Tools::EndUnitTestNo(8);
00285    mtk::Tools::Assert(m11 == m12);
00286 }
00287
00288 void TestConstructAsVandermondeTransposeAssignmentOperator() {
00289
00290    mtk::Tools::BeginUnitTestNo(9);
00291
00292    bool transpose = false;
00293    int gg_l = 3;
00294    int progression_length = 4;
00295    mtk::Real gg[] = {-0.5, 0.5, 1.5};
00296
00297    mtk::DenseMatrix m13(gg, gg_l ,progression_length, transpose);
00298
00299    mtk::DenseMatrix m14;
00300
00301    m14 = m13;
00302
00303    m13.Transpose();
00304
00305    m14 = m13;
00306
```

```
00307   mtk::Tools::EndUnitTestNo(9);
00308   mtk::Tools::Assert(m13 == m14);
00309 }
00310
00311 int main () {
00312
00313   std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00314
00315   TestDefaultConstructor();
00316   TestConstructorWithNumRowsNumCols();
00317   TestConstructAsIdentity();
00318   TestConstructAsVandermonde();
00319   TestSetValueGetValue();
00320   TestConstructAsVandermondeTranspose();
00321   TestKron();
00322   TestConstructWithNumRowsNumColsAssignmentOperator();
00323   TestConstructAsVandermondeTransposeAssignmentOperator();
00324 }
00325
00326 #else
00327 #include <iostream>
00328 using std::cout;
00329 using std::endl;
00330 int main () {
00331   cout << "This code HAS to be compiled with support for C++11." << endl;
00332   cout << "Exiting..." << endl;
00333 }
00334 #endif
```
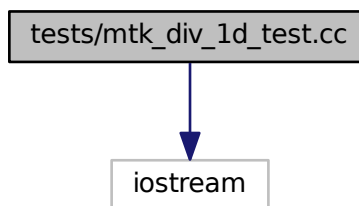
17.119   tests/mtk_div_1d_test.cc File Reference

Testing the mimetic 1D divergence, constructed with the CBS algorithm.

```
#include <iostream>
```
Include dependency graph for mtk_div_1d_test.cc:

Functions

- int main ()

17.119.1   Detailed Description

Author

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

Definition in file mtk_div_1d_test.cc.

17.119.2 Function Documentation

17.119.2.1 int main ( )

Definition at line 288 of file mtk_div_1d_test.cc.

17.120 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
```
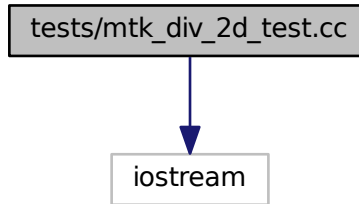
17.121   tests/mtk_div_2d_test.cc File Reference

Test file for the mtk::Div2D class.

```
#include <iostream>
```
Include dependency graph for mtk_div_2d_test.cc:

Functions

- int main ()

17.121.1 Detailed Description

Author

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

Definition in file mtk_div_2d_test.cc.

17.121.2 Function Documentation

17.121.2.1 int main ( )

Definition at line 139 of file mtk_div_2d_test.cc.

17.122 mtk_div_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```

```
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065   mtk::Tools::BeginUnitTestNo(1);
00066
00067   mtk::Div2D dd;
00068
00069   mtk::Real aa = 0.0;
00070   mtk::Real bb = 1.0;
00071   mtk::Real cc = 0.0;
00072   mtk::Real ee = 1.0;
00073
00074   int nn = 5;
00075   int mm = 5;
00076
00077   mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00078
00079   bool assertion = dd.ConstructDiv2D(ddg);
00080
00081   if (!assertion) {
00082     std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00083   }
00084
00085   mtk::Tools::EndUnitTestNo(1);
00086   mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091   mtk::Tools::BeginUnitTestNo(2);
00092
00093   mtk::Div2D dd;
00094
00095   mtk::Real aa = 0.0;
00096   mtk::Real bb = 1.0;
00097   mtk::Real cc = 0.0;
00098   mtk::Real ee = 1.0;
00099
00100   int nn = 5;
00101   int mm = 5;
00102
00103   mtk::UniStgGrid2D ddg(aa, bb, nn, cc, ee, mm);
00104
00105   bool assertion = dd.ConstructDiv2D(ddg);
```

```
00106
00107   if (!assertion) {
00108     std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00109   }
00110
00111   mtk::DenseMatrix ddm(dd.ReturnAsDenseMatrix());
00112
00113   assertion = assertion && (ddm.num_rows() != mtk::kZero);
00114
00115   std::cout << ddm << std::endl;
00116
00117   assertion = assertion && ddm.WriteToFile("mtk_div_2d_test_02.dat");
00118
00119   if(!assertion) {
00120     std::cerr << "Error writing to file." << std::endl;
00121   }
00122
00123   mtk::Tools::EndUnitTestNo(2);
00124   mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129   std::cout << "Testing mtk::Div2D class." << std::endl;
00130
00131   TestDefaultConstructorFactory();
00132   TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140   cout << "This code HAS to be compiled with support for C++11." << endl;
00141   cout << "Exiting..." << endl;
00142 }
00143 #endif
```
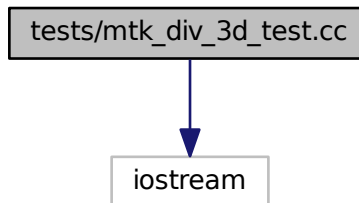
17.123   tests/mtk_div_3d_test.cc File Reference

Test file for the mtk::Div3D class.

```
#include <iostream>
```
Include dependency graph for mtk_div_3d_test.cc:

Functions

- int main ()

17.123.1 Detailed Description

Author

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

Definition in file mtk_div_3d_test.cc.

17.123.2 Function Documentation

17.123.2.1 int main ( )

Definition at line 145 of file mtk_div_3d_test.cc.

17.124 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
```
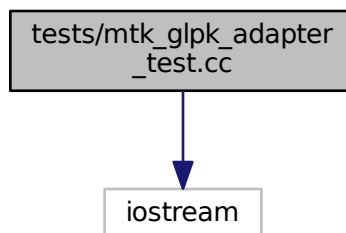
```
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146   cout << "This code HAS to be compiled with support for C++11." << endl;
00147   cout << "Exiting..." << endl;
00148 }
00149 #endif
```

17.125   tests/mtk_glpk_adapter_test.cc File Reference

Test file for the mtk::GLPKAdapter class.

```
#include <iostream>
```
Include dependency graph for mtk_glpk_adapter_test.cc:

Functions

- int main ()

17.125.1   Detailed Description

Author

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

Todo Test the mtk::GLPKAdapter class.

Definition in file mtk_glpk_adapter_test.cc.

17.125.2   Function Documentation

17.125.2.1   int main (   )

Definition at line 81 of file mtk_glpk_adapter_test.cc.
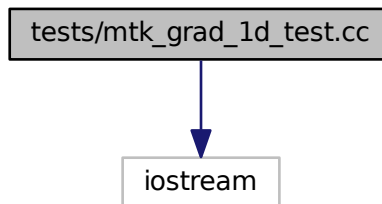
17.126 mtk_glpk_adapter_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065   mtk::Tools::BeginUnitTestNo(1);
00066
00067   mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072   std::cout << "Testing mtk::GLPKAdapter class." << std::endl;
00073
00074   Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082   cout << "This code HAS to be compiled with support for C++11." << endl;
00083   cout << "Exiting..." << endl;
00084 }
00085 #endif
```

17.127 tests/mtk_grad_1d_test.cc File Reference

Testing the mimetic 1D gradient, constructed with the CBS algorithm.

```
#include <iostream>
```
Include dependency graph for mtk_grad_1d_test.cc:

Functions

- int main ()

17.127.1 Detailed Description

Author

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

Definition in file mtk_grad_1d_test.cc.

17.127.2 Function Documentation

17.127.2.1 int main ( )

Definition at line 319 of file mtk_grad_1d_test.cc.

17.128 mtk_grad_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
```

```
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void TestDefaultConstructorFactoryMethodDefault() {
00061
00062   mtk::Tools::BeginUnitTestNo(1);
00063
00064   mtk::Grad1D grad2;
00065
00066   bool assertion = grad2.ConstructGrad1D();
00067
00068   if (!assertion) {
00069     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00070
00071   }
00072
00073   std::cout << grad2 << std::endl;
00074
00075   mtk::Tools::EndUnitTestNo(1);
00076   mtk::Tools::Assert(assertion);
00077 }
00078
00079 void TestDefaultConstructorFactoryMethodFourthOrder() {
00080
00081   mtk::Tools::BeginUnitTestNo(2);
00082
00083   mtk::Grad1D grad4;
00084
00085   bool assertion = grad4.ConstructGrad1D(4);
00086
00087   if (!assertion) {
00088     std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00089   }
00090
00091   std::cout << grad4 << std::endl;
00092
00093   mtk::Tools::EndUnitTestNo(2);
00094   mtk::Tools::Assert(assertion);
00095 }
00096
00097 void TestDefaultConstructorFactoryMethodSixthOrder() {
00098
00099   mtk::Tools::BeginUnitTestNo(3);
```

```
00100
00101   mtk::Grad1D grad6;
00102
00103   bool assertion = grad6.ConstructGrad1D(6);
00104
00105   if (!assertion) {
00106     std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00107   }
00108
00109   std::cout << grad6 << std::endl;
00110
00111   mtk::Tools::EndUnitTestNo(3);
00112   mtk::Tools::Assert(assertion);
00113 }
00114
00115 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00116
00117   mtk::Tools::BeginUnitTestNo(4);
00118
00119   mtk::Grad1D grad8;
00120
00121   bool assertion = grad8.ConstructGrad1D(8);
00122
00123   if (!assertion) {
00124     std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00125   }
00126
00127   std::cout << grad8 << std::endl;
00128
00129   mtk::Tools::EndUnitTestNo(4);
00130   mtk::Tools::Assert(assertion);
00131 }
00132
00133 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00134
00135   mtk::Tools::BeginUnitTestNo(5);
00136
00137   mtk::Grad1D grad10;
00138
00139   bool assertion = grad10.ConstructGrad1D(10);
00140
00141   if (!assertion) {
00142     std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00143   }
00144
00145   std::cout << grad10 << std::endl;
00146
00147   mtk::Tools::EndUnitTestNo(5);
00148   mtk::Tools::Assert(assertion);
00149 }
00150
00151 void TestReturnAsDenseMatrixWithGrid() {
00152
00153   mtk::Tools::BeginUnitTestNo(6);
00154
00155   mtk::Grad1D grad2;
00156
00157   bool assertion = grad2.ConstructGrad1D();
00158
00159   if (!assertion) {
00160     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00161   }
00162
00163   mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00164
00165   mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00166
00167   int rr{6};
00168   int cc{7};
00169
00170   mtk::DenseMatrix ref(rr, cc);
00171
00172   // Row 1.
00173   ref.SetValue(0,0,-13.3333);
00174   ref.SetValue(0,1,15);
00175   ref.SetValue(0,2,-1.66667);
00176   ref.SetValue(0,3,0.0);
00177   ref.SetValue(0,4,0.0);
00178   ref.SetValue(0,5,0.0);
00179   ref.SetValue(0,6,0.0);
00180
```

```
00181    // Row 2.
00182    ref.SetValue(1,0,0.0);
00183    ref.SetValue(1,1,-5.0);
00184    ref.SetValue(1,2,5.0);
00185    ref.SetValue(1,3,0.0);
00186    ref.SetValue(1,4,0.0);
00187    ref.SetValue(1,5,0.0);
00188    ref.SetValue(1,6,0.0);
00189
00190    // Row 3.
00191    ref.SetValue(2,0,0.0);
00192    ref.SetValue(2,1,0.0);
00193    ref.SetValue(2,2,-5.0);
00194    ref.SetValue(2,3,5.0);
00195    ref.SetValue(2,4,0.0);
00196    ref.SetValue(2,5,0.0);
00197    ref.SetValue(2,6,0.0);
00198
00199    // Row 4.
00200    ref.SetValue(3,0,0.0);
00201    ref.SetValue(3,1,0.0);
00202    ref.SetValue(3,2,0.0);
00203    ref.SetValue(3,3,-5.0);
00204    ref.SetValue(3,4,5.0);
00205    ref.SetValue(3,5,0.0);
00206    ref.SetValue(3,6,0.0);
00207
00208    // Row 5.
00209    ref.SetValue(4,0,0.0);
00210    ref.SetValue(4,1,0.0);
00211    ref.SetValue(4,2,0.0);
00212    ref.SetValue(4,3,0.0);
00213    ref.SetValue(4,4,-5.0);
00214    ref.SetValue(4,5,5.0);
00215    ref.SetValue(4,6,0.0);
00216
00217    // Row 6.
00218    ref.SetValue(5,0,0.0);
00219    ref.SetValue(5,1,0.0);
00220    ref.SetValue(5,2,0.0);
00221    ref.SetValue(5,3,0.0);
00222    ref.SetValue(5,4,1.66667);
00223    ref.SetValue(5,5,-15.0);
00224    ref.SetValue(5,6,13.3333);
00225
00226    mtk::Tools::EndUnitTestNo(6);
00227    mtk::Tools::Assert(grad2m == ref);
00228 }
00229
00230 void TestReturnAsDimensionlessDenseMatrix() {
00231
00232    mtk::Tools::BeginUnitTestNo(7);
00233
00234    mtk::Grad1D grad4;
00235
00236    bool assertion = grad4.ConstructGrad1D(4);
00237
00238    if (!assertion) {
00239      std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00240    }
00241
00242    mtk::DenseMatrix grad4m(grad4.ReturnAsDimensionlessDenseMatrix
      (10));
00243
00244    std::cout << grad4m << std::endl;
00245
00246    mtk::Tools::EndUnitTestNo(7);
00247    mtk::Tools::Assert(assertion);
00248 }
00249
00250 void TestWriteToFile() {
00251
00252    mtk::Tools::BeginUnitTestNo(8);
00253
00254    mtk::Grad1D grad2;
00255
00256    bool assertion = grad2.ConstructGrad1D();
00257
00258    if (!assertion) {
00259      std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00260    }
```

```
00261
00262    mtk::UniStgGrid1D grid(0.0, 1.0, 50);
00263
00264    mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00265
00266    std::cout << grad2m << std::endl;
00267
00268    assertion = assertion && grad2m.WriteToFile("mtk_grad_1d_test_08.dat");
00269
00270    if(!assertion) {
00271      std::cerr << "Error writing to file." << std::endl;
00272    }
00273
00274    mtk::Tools::EndUnitTestNo(8);
00275    mtk::Tools::Assert(assertion);
00276 }
00277
00278 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
```
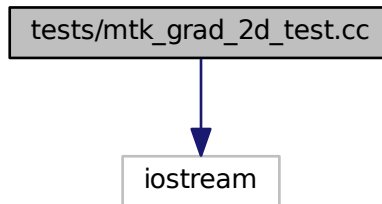
17.129 tests/mtk_grad_2d_test.cc File Reference

Test file for the mtk::Grad2D class.

```
#include <iostream>
```
Include dependency graph for mtk_grad_2d_test.cc:

Functions

- int main ()

17.129.1 Detailed Description

Author

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

Definition in file mtk_grad_2d_test.cc.

17.129.2 Function Documentation

17.129.2.1 int main ( )

Definition at line 139 of file mtk_grad_2d_test.cc.

17.130 mtk_grad_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```

```
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065   mtk::Tools::BeginUnitTestNo(1);
00066
00067   mtk::Grad2D gg;
00068
00069   mtk::Real aa = 0.0;
00070   mtk::Real bb = 1.0;
00071   mtk::Real cc = 0.0;
00072   mtk::Real dd = 1.0;
00073
00074   int nn = 5;
00075   int mm = 5;
00076
00077   mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00078
00079   bool assertion = gg.ConstructGrad2D(ggg);
00080
00081   if (!assertion) {
00082     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00083   }
00084
00085   mtk::Tools::EndUnitTestNo(1);
00086   mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091   mtk::Tools::BeginUnitTestNo(2);
00092
00093   mtk::Grad2D gg;
00094
00095   mtk::Real aa = 0.0;
00096   mtk::Real bb = 1.0;
00097   mtk::Real cc = 0.0;
00098   mtk::Real dd = 1.0;
00099
00100   int nn = 5;
00101   int mm = 5;
00102
00103   mtk::UniStgGrid2D ggg(aa, bb, nn, cc, dd, mm, mtk::VECTOR);
00104
00105   bool assertion = gg.ConstructGrad2D(ggg);
```

```
00106
00107   if (!assertion) {
00108     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00109   }
00110
00111   mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00112
00113   assertion = assertion && (ggm.num_rows() != mtk::kZero);
00114
00115   std::cout << ggm << std::endl;
00116
00117   assertion = assertion && ggm.WriteToFile("mtk_grad_2d_test_02.dat");
00118
00119   if(!assertion) {
00120     std::cerr << "Error writing to file." << std::endl;
00121   }
00122
00123   mtk::Tools::EndUnitTestNo(2);
00124   mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129   std::cout << "Testing mtk::Grad2D class." << std::endl;
00130
00131   TestDefaultConstructorFactory();
00132   TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140   cout << "This code HAS to be compiled with support for C++11." << endl;
00141   cout << "Exiting..." << endl;
00142 }
00143 #endif
```
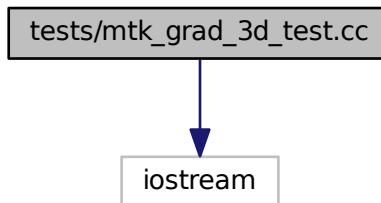
17.131 tests/mtk_grad_3d_test.cc File Reference

Test file for the mtk::Grad3D class.

```
#include <iostream>
```
Include dependency graph for mtk_grad_3d_test.cc:

Functions

- int main ()

17.131.1 Detailed Description

Author

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

Definition in file mtk_grad_3d_test.cc.

17.131.2 Function Documentation

17.131.2.1 int main ( )

Definition at line 145 of file mtk_grad_3d_test.cc.

17.132 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,
      mtk::VECTOR);
00081
00082   bool assertion = gg.ConstructGrad3D(ggg);
00083
00084   if (!assertion) {
00085     std::cerr << "Mimetic grad (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::Grad3D gg;
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 ggg(aa, bb, nn, cc, dd, mm, ee, ff, oo,
      mtk::VECTOR);
00110
00111   bool assertion = gg.ConstructGrad3D(ggg);
00112
00113   if (!assertion) {
00114     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00115   }
00116
00117   mtk::DenseMatrix ggm(gg.ReturnAsDenseMatrix());
00118
00119   assertion = assertion && (ggm.num_rows() != mtk::kZero);
00120
00121   std::cout << ggm << std::endl;
00122
00123   assertion = assertion && ggm.WriteToFile("mtk_grad_3d_test_02.dat");
00124
00125   if(!assertion) {
00126     std::cerr << "Error writing to file." << std::endl;
00127   }
00128
00129   mtk::Tools::EndUnitTestNo(2);
00130   mtk::Tools::Assert(assertion);
00131 }
00132
00133 int main () {
00134
00135   std::cout << "Testing mtk::Grad2D class." << std::endl;
00136
00137   TestDefaultConstructorFactory();
00138   TestReturnAsDenseMatrixWriteToFile();
00139 }
```
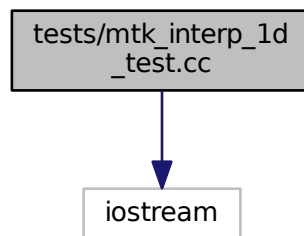
```
00140
00141 #else
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146   cout << "This code HAS to be compiled with support for C++11." << endl;
00147   cout << "Exiting..." << endl;
00148 }
00149 #endif
```

17.133 tests/mtk_interp_1d_test.cc File Reference

Testing the 1D interpolation.

`#include <iostream>`
Include dependency graph for mtk_interp_1d_test.cc:

Functions

- int main ()

17.133.1 Detailed Description

Testing the 1D interpolation.

Author

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

Definition in file mtk_interp_1d_test.cc.

17.133.2 Function Documentation

17.133.2.1 int main ( )

Definition at line 113 of file mtk_interp_1d_test.cc.

17.134 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
```
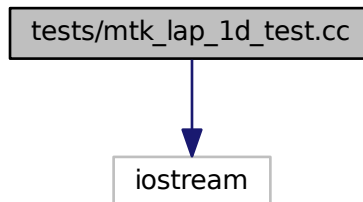
17.135 tests/mtk_lap_1d_test.cc File Reference

Testing the 1D Laplacian operator.

```
#include <iostream>
```
Include dependency graph for mtk_lap_1d_test.cc:

Functions

- int main ()

17.135.1 Detailed Description

Author

> : Eduardo J. Sanchez (ejspeiro) - esanchez at mail dot sdsu dot edu
>
> : Johnny Corbino - jcorbino at mail dot sdsu dot edu

Definition in file mtk_lap_1d_test.cc.

17.135.2 Function Documentation

17.135.2.1 int main ( )

Definition at line 193 of file mtk_lap_1d_test.cc.

17.136 mtk_lap_1d_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059
00060 #include "mtk.h"
00061
00062 void TestDefaultConstructorFactoryMethodDefault() {
00063
00064   mtk::Tools::BeginUnitTestNo(1);
```

```
00065
00066   mtk::Lap1D lap2;
00067
00068   bool assertion = lap2.ConstructLap1D();
00069
00070   if (!assertion) {
00071     std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00072   }
00073
00074   mtk::Tools::EndUnitTestNo(1);
00075   mtk::Tools::Assert(assertion);
00076 }
00077
00078 void TestDefaultConstructorFactoryMethodFourthOrder() {
00079
00080   mtk::Tools::BeginUnitTestNo(2);
00081
00082   mtk::Lap1D lap4;
00083
00084   bool assertion = lap4.ConstructLap1D(4);
00085
00086   if (!assertion) {
00087     std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00088   }
00089
00090   mtk::Tools::EndUnitTestNo(2);
00091   mtk::Tools::Assert(assertion);
00092 }
00093
00094 void TestDefaultConstructorFactoryMethodSixthOrder() {
00095
00096   mtk::Tools::BeginUnitTestNo(3);
00097
00098   mtk::Lap1D lap6;
00099
00100   bool assertion = lap6.ConstructLap1D(6);
00101
00102   if (!assertion) {
00103     std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00104   }
00105
00106   mtk::Tools::EndUnitTestNo(3);
00107   mtk::Tools::Assert(assertion);
00108 }
00109
00110 void TestDefaultConstructorFactoryMethodEightOrderDefThreshold() {
00111
00112   mtk::Tools::BeginUnitTestNo(4);
00113
00114   mtk::Lap1D lap8;
00115
00116   bool assertion = lap8.ConstructLap1D(8);
00117
00118   if (!assertion) {
00119     std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00120   }
00121
00122   mtk::Tools::EndUnitTestNo(4);
00123 }
00124
00125 void TestDefaultConstructorFactoryMethodTenthOrderDefThreshold() {
00126
00127   mtk::Tools::BeginUnitTestNo(5);
00128
00129   mtk::Lap1D lap10;
00130
00131   bool assertion = lap10.ConstructLap1D(10);
00132
00133   if (!assertion) {
00134     std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00135   }
00136
00137   mtk::Tools::EndUnitTestNo(5);
00138 }
00139
00140 void TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold() {
00141
00142   mtk::Tools::BeginUnitTestNo(6);
00143
00144   mtk::Lap1D lap12;
00145
```

```
00146    bool assertion = lap12.ConstructLap1D(12);
00147
00148    if (!assertion) {
00149      std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00150    }
00151
00152    mtk::Tools::EndUnitTestNo(6);
00153 }
00154
00155 void TestReturnAsDenseMatrix() {
00156
00157    mtk::Tools::BeginUnitTestNo(8);
00158
00159    mtk::Lap1D lap4;
00160
00161    bool assertion = lap4.ConstructLap1D(4);
00162
00163    if (!assertion) {
00164      std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00165    }
00166
00167    mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00168
00169    mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00170
00171    assertion = assertion &&
00172        abs(lap4_m.GetValue(1, 0) - 385.133) < mtk::kDefaultTolerance &&
00173        abs(lap4_m.GetValue(11, 12) - 385.133) < mtk::kDefaultTolerance;
00174    mtk::Tools::EndUnitTestNo(8);
00175    mtk::Tools::Assert(assertion);
00176 }
00177
00178 int main () {
00179
00180    std::cout << "Testing MTK 1D Laplacian" << std::endl;
00181
00182    TestDefaultConstructorFactoryMethodDefault();
00183    TestDefaultConstructorFactoryMethodFourthOrder();
00184    TestDefaultConstructorFactoryMethodSixthOrder();
00185    TestDefaultConstructorFactoryMethodEightOrderDefThreshold();
00186    TestDefaultConstructorFactoryMethodTenthOrderDefThreshold();
00187    TestDefaultConstructorFactoryMethodTwelfthOrderDefThreshold();
00188    TestReturnAsDenseMatrix();
00189 }
00190
00191 #else
00192 #include <iostream>
00193 int main () {
00194    std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00195    std::cout << "Exiting..." << std::endl;
00196 }
00197 #endif
```
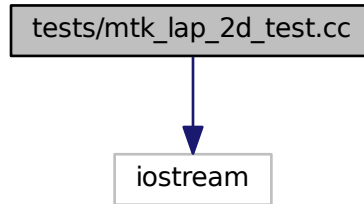
17.137   tests/mtk_lap_2d_test.cc File Reference

Test file for the mtk::Lap2D class.

```
#include <iostream>
```
Include dependency graph for mtk_lap_2d_test.cc:

Functions

- int main ()

17.137.1 Detailed Description

Author

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

Definition in file mtk_lap_2d_test.cc.

17.137.2 Function Documentation

17.137.2.1 int main ( )

Definition at line 139 of file mtk_lap_2d_test.cc.

17.138 mtk_lap_2d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
```

```
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <cmath>
00057 #include <ctime>
00058
00059 #include <iostream>
00060
00061 #include "mtk.h"
00062
00063 void TestDefaultConstructorFactory() {
00064
00065   mtk::Tools::BeginUnitTestNo(1);
00066
00067   mtk::Lap2D ll;
00068
00069   mtk::Real aa = 0.0;
00070   mtk::Real bb = 1.0;
00071   mtk::Real cc = 0.0;
00072   mtk::Real dd = 1.0;
00073
00074   int nn = 5;
00075   int mm = 5;
00076
00077   mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00078
00079   bool assertion = ll.ConstructLap2D(llg);
00080
00081   if (!assertion) {
00082     std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00083   }
00084
00085   mtk::Tools::EndUnitTestNo(1);
00086   mtk::Tools::Assert(assertion);
00087 }
00088
00089 void TestReturnAsDenseMatrixWriteToFile() {
00090
00091   mtk::Tools::BeginUnitTestNo(2);
00092
00093   mtk::Lap2D ll;
00094
00095   mtk::Real aa = 0.0;
00096   mtk::Real bb = 1.0;
00097   mtk::Real cc = 0.0;
00098   mtk::Real dd = 1.0;
00099
00100   int nn = 5;
00101   int mm = 5;
00102
00103   mtk::UniStgGrid2D llg(aa, bb, nn, cc, dd, mm);
00104
00105   bool assertion = ll.ConstructLap2D(llg);
```

```
00106
00107   if (!assertion) {
00108     std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00109   }
00110
00111   mtk::DenseMatrix llm(ll.ReturnAsDenseMatrix());
00112
00113   assertion = assertion && (llm.num_rows() != 0);
00114
00115   std::cout << llm << std::endl;
00116
00117   assertion = assertion && llm.WriteToFile("mtk_lap_2d_test_02.dat");
00118
00119   if(!assertion) {
00120     std::cerr << "Error writing to file." << std::endl;
00121   }
00122
00123   mtk::Tools::EndUnitTestNo(2);
00124   mtk::Tools::Assert(assertion);
00125 }
00126
00127 int main () {
00128
00129   std::cout << "Testing mtk::Lap2D class." << std::endl;
00130
00131   TestDefaultConstructorFactory();
00132   TestReturnAsDenseMatrixWriteToFile();
00133 }
00134
00135 #else
00136 #include <iostream>
00137 using std::cout;
00138 using std::endl;
00139 int main () {
00140   cout << "This code HAS to be compiled with support for C++11." << endl;
00141   cout << "Exiting..." << endl;
00142 }
00143 #endif
```
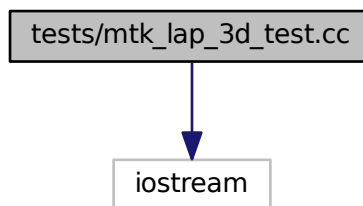
17.139   tests/mtk_lap_3d_test.cc File Reference

Test file for the mtk::Lap3D class.

```
#include <iostream>
```
Include dependency graph for mtk_lap_3d_test.cc:

Functions

- int main ()

17.139.1 Detailed Description

Author

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

Definition in file mtk_lap_3d_test.cc.

17.139.2 Function Documentation

17.139.2.1 int main ( )

Definition at line 145 of file mtk_lap_3d_test.cc.

17.140 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
```
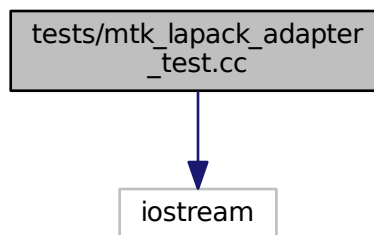
```
00142 #include <iostream>
00143 using std::cout;
00144 using std::endl;
00145 int main () {
00146   cout << "This code HAS to be compiled with support for C++11." << endl;
00147   cout << "Exiting..." << endl;
00148 }
00149 #endif
```

17.141 tests/mtk_lapack_adapter_test.cc File Reference

Test file for the mtk::LAPACKAdapter class.

`#include <iostream>`
Include dependency graph for mtk_lapack_adapter_test.cc:

Functions

- int main ()

17.141.1 Detailed Description

Author

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

Todo Test the mtk::LAPACKAdapter class.

Definition in file mtk_lapack_adapter_test.cc.

17.141.2 Function Documentation

17.141.2.1 int main ( )

Definition at line 81 of file mtk_lapack_adapter_test.cc.

17.142 mtk_lapack_adapter_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed, unless these modifications are made through the project's GitHub
00020 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00021 should be developed and included in any deliverable.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions in binary form must reproduce the above copyright notice,
00027 this list of conditions and the following disclaimer in the documentation and/or
00028 other materials provided with the distribution.
00029
00030 4. Usage of the binary form on proprietary applications shall require explicit
00031 prior written permission from the the copyright holders, and due credit should
00032 be given to the copyright holders.
00033
00034 5. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065   mtk::Tools::BeginUnitTestNo(1);
00066
00067   mtk::Tools::EndUnitTestNo(1);
00068 }
00069
00070 int main () {
00071
00072   std::cout << "Testing mtk::LAPACKAdapter class." << std::endl;
00073
00074   Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082   cout << "This code HAS to be compiled with support for C++11." << endl;
00083   cout << "Exiting..." << endl;
00084 }
00085 #endif
```
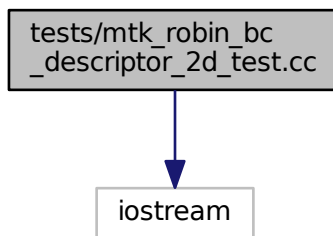
17.143 tests/mtk_robin_bc_descriptor_2d_test.cc File Reference

Test file for the mtk::RobinBCDescriptor2D class.

```
#include <iostream>
```
Include dependency graph for mtk_robin_bc_descriptor_2d_test.cc:

Functions

- int main ()

17.143.1 Detailed Description

Author

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

Definition in file mtk_robin_bc_descriptor_2d_test.cc.

17.143.2 Function Documentation

17.143.2.1 int main ( )

Definition at line 198 of file mtk_robin_bc_descriptor_2d_test.cc.

17.144 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
00187   std::cout << "Testing mtk::RobinBCDescriptor2D class." << std::endl;
00188
00189   TestDefaultConstructorGetters();
00190   TestPushBackImposeOnLaplacianMatrix();
00191   TestImposeOnGrid();
00192 }
00193
00194 #else
00195 #include <iostream>
00196 using std::cout;
00197 using std::endl;
00198 int main () {
00199   cout << "This code HAS to be compiled with support for C++11." << endl;
00200   cout << "Exiting..." << endl;
00201 }
00202 #endif
```
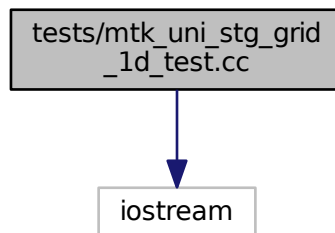
17.145   tests/mtk_uni_stg_grid_1d_test.cc File Reference

Test file for the mtk::UniStgGrid1D class.

```
#include <iostream>
```
Include dependency graph for mtk_uni_stg_grid_1d_test.cc:

Functions

- int main ()

17.145.1   Detailed Description

Author

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

Definition in file mtk_uni_stg_grid_1d_test.cc.

17.145.2 Function Documentation

17.145.2.1 int main ( )

Definition at line 172 of file mtk_uni_stg_grid_1d_test.cc.

17.146 mtk_uni_stg_grid_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed, unless these modifications are made through the project's GitHub
00018 page: http://www.csrc.sdsu.edu/mtk. Documentation related to said modifications
00019 should be developed and included in any deliverable.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions in binary form must reproduce the above copyright notice,
00025 this list of conditions and the following disclaimer in the documentation and/or
00026 other materials provided with the distribution.
00027
00028 4. Usage of the binary form on proprietary applications shall require explicit
00029 prior written permission from the the copyright holders, and due credit should
00030 be given to the copyright holders.
00031
00032 5. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void TestDefaultConstructor() {
00062
00063   mtk::Tools::BeginUnitTestNo(1);
00064
00065   mtk::UniStgGrid1D gg;
00066
00067   mtk::Tools::EndUnitTestNo(1);
00068   mtk::Tools::Assert(gg.delta_x() == mtk::kZero);
00069 }
00070
00071 mtk::Real ScalarField(const mtk::Real &xx) {
00072
00073   return 2.0*xx;
```

```
00074 }
00075
00076 void TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField() {
00077
00078   mtk::Tools::BeginUnitTestNo(2);
00079
00080   mtk::Real aa = 0.0;
00081   mtk::Real bb = 1.0;
00082
00083   int nn = 5;
00084
00085   mtk::UniStgGrid1D gg(aa, bb, nn);
00086
00087   gg.BindScalarField(ScalarField);
00088
00089   std::cout << gg << std::endl;
00090
00091   mtk::Tools::EndUnitTestNo(2);
00092   mtk::Tools::Assert(gg.delta_x() == 0.2 && gg.
    num_cells_x() == 5);
00093 }
00094
00095 void TestBindScalarFieldWriteToFile() {
00096
00097   mtk::Tools::BeginUnitTestNo(3);
00098
00099   mtk::Real aa = 0.0;
00100   mtk::Real bb = 1.0;
00101
00102   int nn = 5;
00103
00104   mtk::UniStgGrid1D gg(aa, bb, nn);
00105
00106   bool assertion{true};
00107
00108   gg.BindScalarField(ScalarField);
00109
00110   assertion =
00111     assertion &&
00112     gg.discrete_field()[0] == 0.0 &&
00113     gg.discrete_field()[gg.num_cells_x() + 2 - 1] == 2.0;
00114
00115   if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_03.dat", "x", "u(x)")) {
00116     std::cerr << "Error writing to file." << std::endl;
00117     assertion = false;
00118   }
00119
00120   mtk::Tools::EndUnitTestNo(3);
00121   mtk::Tools::Assert(assertion);
00122 }
00123
00124 mtk::Real VectorFieldPComponent(mtk::Real xx) {
00125
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::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   TestConstructWithWestBndyEastBndyNumCellsOStreamOperatorBindScalarField();
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
```
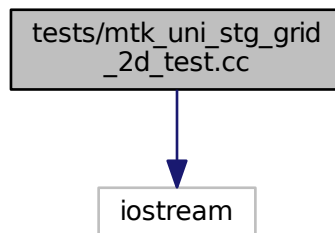
17.147   tests/mtk_uni_stg_grid_2d_test.cc File Reference

Test file for the mtk::UniStgGrid2D class.

```
#include <iostream>
```
Include dependency graph for mtk_uni_stg_grid_2d_test.cc:

Functions

- int main ()

17.147.1   Detailed Description

Author

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

Definition in file mtk_uni_stg_grid_2d_test.cc.

17.147.2   Function Documentation

17.147.2.1   int main (   )

Definition at line 202 of file mtk_uni_stg_grid_2d_test.cc.

17.148   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.
      delta_y() == mtk::kZero);
00071 }
00072
```

```
00073 void
00074 TestConstructWithWestEastNumCellsXSouthNorthBndysNumCellsYOStreamOperator() {
00075
00076   mtk::Tools::BeginUnitTestNo(2);
00077
00078   mtk::Real aa = 0.0;
00079   mtk::Real bb = 1.0;
00080   mtk::Real cc = 0.0;
00081   mtk::Real dd = 1.0;
00082
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) <
     mtk::kDefaultTolerance);
00093 }
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::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
```
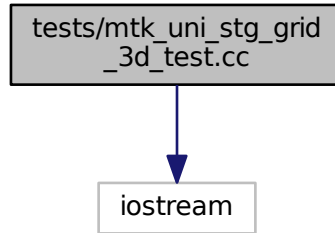
17.149 tests/mtk_uni_stg_grid_3d_test.cc File Reference

Test file for the mtk::UniStgGrid3D class.

```
#include <iostream>
```
Include dependency graph for mtk_uni_stg_grid_3d_test.cc:

Functions

- int main ()

17.149.1 Detailed Description

Author

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

Definition in file mtk_uni_stg_grid_3d_test.cc.

17.149.2 Function Documentation

17.149.2.1 int main ( )

Definition at line 184 of file mtk_uni_stg_grid_3d_test.cc.

17.150 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);
00098 }
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