MTK: Mimetic Methods Toolkit

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

1.1   MTK Concerns

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

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

In order of dependence these are:

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

1.2   MTK Flavors

The MTK collection of wrappers is:

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

Others are being designed and developed.

1.3 Contact, Support and Credits

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

Developers are members of:

1. Mimetic Numerical Methods Research and Development Group.

2. Computational Geoscience Research and Development Group.

3. Ocean Modeling Research and Development Group.

Currently the developers are:

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

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

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

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

5. Angel Boada.

6. Johnny Corbino.

7. Raul Vargas-Navarro.

1.4 Acknowledgements and Contributions

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

1. Mohammad Abouali, Ph.D.

2. Dany De Cecchis, Ph.D.

3. Julia Rossi.

Chapter 2

Programming Tools

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

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

2. Compiler: gcc version 4.4.5 (Ubuntu/Linaro 4.4.4-14ubuntu5). Copyright (C) 2013 Free Software Foundation, Inc.

3. Debugger: GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1. Copyright (C) 2014 Free Software Foundation, Inc.

Chapter 3

Licensing and Modifications

Chapter 4

Read Me File and Installation Instructions

README File for the Mimetic Methods Toolkit (MTK)

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

1. Description

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

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

2. Dependencies

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

$(HOME)/Libraries/

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

For Linux:

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

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

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

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

For OS X:

```
There are no dependences for OS X.
```

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:
- make: builds only the library and the examples.
- all: builds the library, the examples and the documentation.
- mtklib: builds the library, i.e. generates the archive files.
```

```
- test: generates the tests.
- example: generates the examples.
- gendoc: generates the documentation for the library.

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

PART 2. BUILD THE LIBRARY.

```
$ make
```

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

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

```
Examples and tests will also be built.
```

4. Frequently Asked Questions

```
Q: Why haven't you guys implemented GBS to build the library?
A: I'm on it as we speak! ;)

Q: Is there any main reference when it comes to the theory on Mimetic Methods?
A: Yes! Check:  http://www.csrc.sdsu.edu/mimetic-book

Q: Do I need to generate the documentation myself?
A: You can if you want to... but if you DO NOT want to, just go to our website.
```

5. Contact, Support, and Credits

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

```
Developers are members of:
```

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

Currently the developers are:

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

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

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

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

Thanks and happy coding!

Chapter 5

Tests and Test Architectures

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

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

```
1. Linux 3.2.0-23-generic-pae #36-Ubuntu SMP i386 GNU/Linux
   Intel(R) Pentium(R) M processor 1.73GHz 2048 KB of cache and stepping of 8
   gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
```

Further architectures will be tested!

Chapter 6

Examples

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

Chapter 7

User Manual, References and Theory

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

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

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

Chapter 8

Todo List

Member mtk::DenseMatrix::OrderColMajor ()

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

Member mtk::DenseMatrix::OrderRowMajor ()

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

Member mtk::DenseMatrix::Transpose ()

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

Class mtk::GLPKAdapter

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

Member mtk::Matrix::IncreaseNumNull ()

   Review the definition of sparse matrices properties.

Member mtk::Matrix::IncreaseNumZero ()

   Review the definition of sparse matrices properties.

Member mtk::Tools::Prevent (const bool condition, const char ∗fname, int lineno, const char ∗fxname)

   Check if this is the best way of stalling execution.

Member mtk::Tools::test_number_

   Check usage of static methods and private members.

File mtk_dense_matrix.h

   Add sparse matrices support: BANDED and CRS.

   Contemplate manipulation of sparse metrics.

   Implement Kronecker product using the BLAS.

File mtk_div_1d.cc

   Overload ostream operator as in mtk::Lap1D.

   Implement creation of ■ w. mtk::BLASAdapter.

File mtk_glpk_adapter.cc

   Document better this file.

File mtk_glpk_adapter_test.cc

   Test the mtk::GLPKAdapter class.

File mtk_grad_1d.cc

   Overload ostream operator as in mtk::Lap1D.

   Implement creation of ■ w. mtk::BLASAdapter.

File mtk_lapack_adapter_test.cc

    Test the mtk::LAPACKAdapter class.

File mtk_quad_1d.h

    Implement this class.

File mtk_roots.h

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

    Test selective precision mechanism.

File mtk_uni_stg_grid_1d.h

    Create overloaded binding routines that read data from files.

Chapter 9

Bug List

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

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

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

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

Chapter 10

Module Index

10.1   Modules

Here is a list of all modules:

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::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 {1.e-6f}

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

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

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

14.1.1 Detailed Description

Fundamental execution parameters and defined types.

14.1.2 Typedef Documentation

14.1.2.1 mtk::Real

Definition at line 83 of file mtk_roots.h.

14.1.3 Variable Documentation

14.1.3.1 mtk::kCriticalOrderAccuracyDiv {8}

Definition at line 157 of file mtk_roots.h.

14.1.3.2 mtk::kCriticalOrderAccuracyGrad {10}

Definition at line 166 of file mtk_roots.h.

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

Warning

> Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 147 of file mtk_roots.h.

14.1.3.4 mtk::kDefaultOrderAccuracy {2}

Warning

> Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 133 of file mtk_roots.h.

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

Definition at line 121 of file mtk_roots.h.

14.1.3.6 mtk::kOne {1.0f}

Warning

> Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 108 of file mtk_roots.h.

14.1.3.7 mtk::kZero {0.0f}

Warning

> Declared as double if MTK_PRECISION_DOUBLE is defined.

Definition at line 107 of file mtk_roots.h.

14.2 Enumerations.

Enumerations.

Enumerations

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

    Considered matrix storage schemes to implement sparse matrices.

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

    Considered matrix ordering (for Fortran purposes).

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

    Nature of the field discretized in a given grid.

14.2.1 Detailed Description

Enumerations.

14.2.2 Enumeration Type Documentation

14.2.2.1 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.

14.2.2.2 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.

14.2.2.3   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.

14.6.1 Detailed Description

Uniform rectangular staggered grids.

14.7 Mimetic operators.

Mimetic operators.

Classes

- class mtk::Div1D

    Implements a 1D mimetic divergence operator.
- class mtk::Grad1D

    Implements a 1D mimetic gradient operator.
- class mtk::Lap1D

    Implements a 1D mimetic Laplacian operator.
- class mtk::Quad1D

    Implements a 1D mimetic quadrature.

14.7.1 Detailed Description

Mimetic operators.

Chapter 15

Namespace Documentation

15.1   mtk Namespace Reference

Mimetic Methods Toolkit namespace.

Classes

- class BLASAdapter

    Adapter class for the BLAS API.
- class DenseMatrix

    Defines a common dense matrix, using a 1D array.
- class Div1D

    Implements a 1D mimetic divergence operator.
- class GLPKAdapter

    Adapter class for the GLPK API.
- class Grad1D

    Implements a 1D mimetic gradient operator.
- class Lap1D

    Implements a 1D 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 Tools

    Tool manager class.
- class UniStgGrid1D

    Uniform 1D Staggered Grid.

Typedefs

- typedef float Real

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

Enumerations

- enum MatrixStorage { DENSE, BANDED, CRS }

    Considered matrix storage schemes to implement sparse matrices.
- enum MatrixOrdering { ROW_MAJOR, COL_MAJOR }

    Considered matrix ordering (for Fortran purposes).
- enum FieldNature { SCALAR, VECTOR }

    Nature of the field discretized in a given grid.

Functions

- float snrm2_ (int ∗n, Real ∗x, int ∗incx)
- void sgemv_ (char ∗trans, int ∗m, int ∗n, Real ∗alpha, Real ∗a, int ∗lda, Real ∗x, int ∗incx, Real ∗beta, Real ∗y, int ∗incy)
- void sgemm_ (char ∗transa, char ∗transb, int ∗m, int ∗n, int ∗k, Real ∗alpha, Real ∗a, int ∗lda, Real ∗b, aamm int ∗ldb, Real ∗beta, Real ∗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::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)

Variables

- const float kZero {0.0f}

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

    MTK's one defined according to selective compilation.
- const float kDefaultTolerance {1e-7f}

    Considered tolerance for comparisons in numerical methods.
- const int kDefaultOrderAccuracy {2}

    Default order of accuracy for mimetic operators.
- const float kDefaultMimeticThreshold {1.e-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::UniStgGrid1D & *in*)

1. Print spatial coordinates.

Definition at line 68 of file mtk_uni_stg_grid_1d.cc.

15.1.1.2 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.

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

Definition at line 75 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
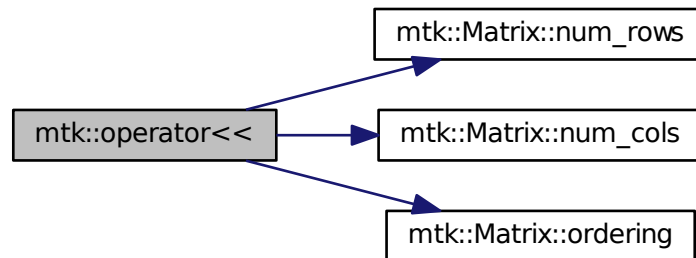
15.1.1.4 std::ostream& mtk::operator$<<$ (std::ostream & *stream,* mtk::Div1D & *in*)

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.

15.1.1.5 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.

15.1.1.6 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:
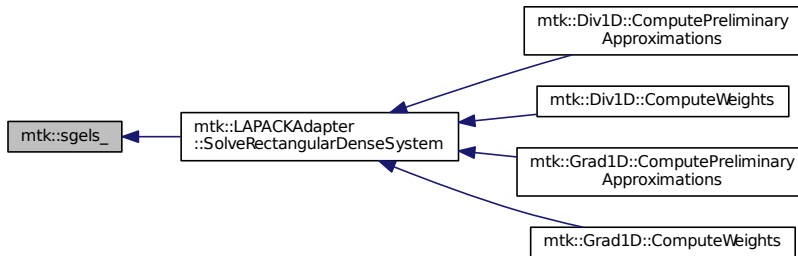
15.1.1.7 **void mtk::sgemm_ (char ∗ *transa,* char ∗ *transb,* int ∗ *m,* int ∗ *n,* int ∗ *k,* Real ∗ *alpha,* Real ∗ *a,* int ∗ *lda,* Real ∗ *b,* aamm int ∗ *ldb,* Real ∗ *beta,* Real ∗ *c,* int ∗ *ldc*)**

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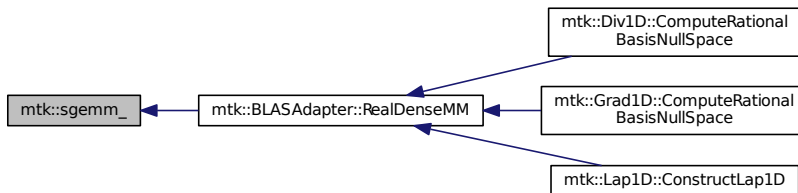
15.1.1.8 **void mtk::sgemv_ (char ∗ *trans,* int ∗ *m,* int ∗ *n,* Real ∗ *alpha,* Real ∗ *a,* int ∗ *lda,* Real ∗ *x,* int ∗ *incx,* Real ∗ *beta,* Real ∗ *y,* int ∗ *incy*)**

Here is the caller graph for this function:
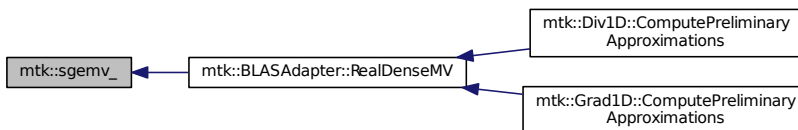
15.1.1.9   void mtk::sgeqrf_ (int ∗ _m,_ int ∗ _n,_ Real ∗ _a,_ int ∗ _lda,_ Real ∗ _tau,_ Real ∗ _work,_ int ∗ _lwork,_ int ∗ _info_)

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.

15.1.1.10   void mtk::sgesv_ (int ∗ _n,_ int ∗ _nrhs,_ Real ∗ _a,_ int ∗ _lda,_ int ∗ _ipiv,_ Real ∗ _b,_ int ∗ _ldb,_ int ∗ _info_)

15.1.1.11   float mtk::snrm2_ (int ∗ _n,_ Real ∗ _x,_ int ∗ _incx_)

Here is the caller graph for this function:
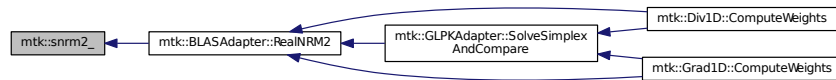
15.1.1.12   void mtk::sormqr_ (char ∗ _side,_ char ∗ _trans,_ int ∗ _m,_ int ∗ _n,_ int ∗ _k,_ Real ∗ _a,_ int ∗ _lda,_ Real ∗ _tau,_ Real ∗ _c,_ int ∗ _ldc,_ Real ∗ _work,_ int ∗ _lwork,_ int ∗ _info_)

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 $\geq$ 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()            │
│ + RealDenseMV()         │
│ + RealDenseMM()         │
└─────────────────────────┘
```
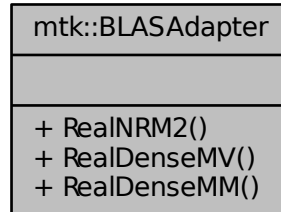
Static Public Member Functions

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

    Compute the $|\mathbf{x}|_2$ of given array $\mathbf{x}$.
- static void **RealDenseMV** (Real &alpha, DenseMatrix &aa, Real ∗xx, Real &beta, Real ∗yy)

    Real-Arithmetic General (Dense matrices) Matrix-Vector Multiplier.
- static DenseMatrix **RealDenseMM** (DenseMatrix &aa, DenseMatrix &bb)

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

16.1.1   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 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/

Definition at line 318 of file mtk_blas_adapter.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

```
                                     ┌─────────────────────────┐
                                     │  mtk::Div1D::ComputeRational │
                                     │      BasisNullSpace      │
                                     └─────────────────────────┘
┌──────────────────────────┐        ┌─────────────────────────┐
│ mtk::BLASAdapter::RealDenseMM │ ◄─── │  mtk::Grad1D::ComputeRational │
└──────────────────────────┘        │      BasisNullSpace      │
                                     └─────────────────────────┘
                                     ┌─────────────────────────┐
                                     │  mtk::Lap1D::ConstructLap1D │
                                     └─────────────────────────┘
```
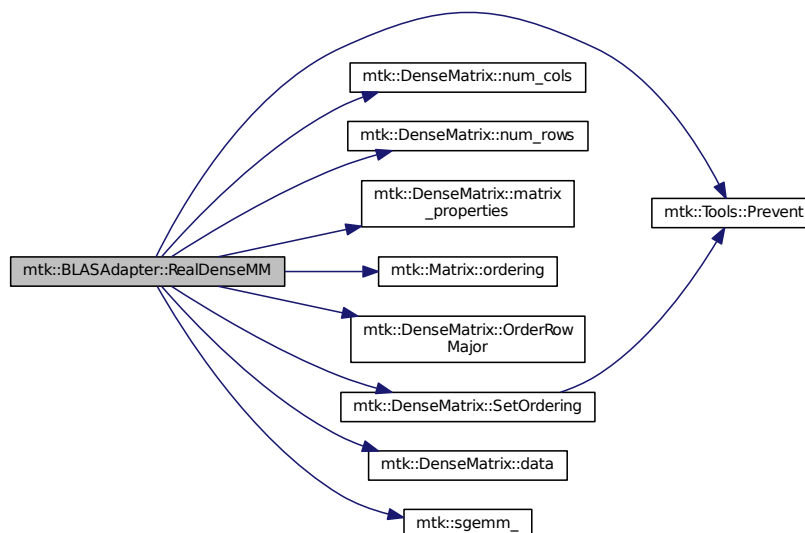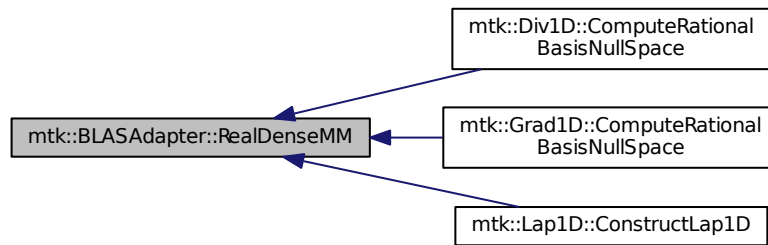
16.1.2.2 **void mtk::BLASAdapter::RealDenseMV (mtk::Real &** *alpha,* **mtk::DenseMatrix &** *aa,* **mtk::Real** $*$ *xx,* **mtk::Real &** *beta,* **mtk::Real** $*$ *yy* **)** `[static]`

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 287 of file mtk_blas_adapter.cc.

Here is the call graph for this function:
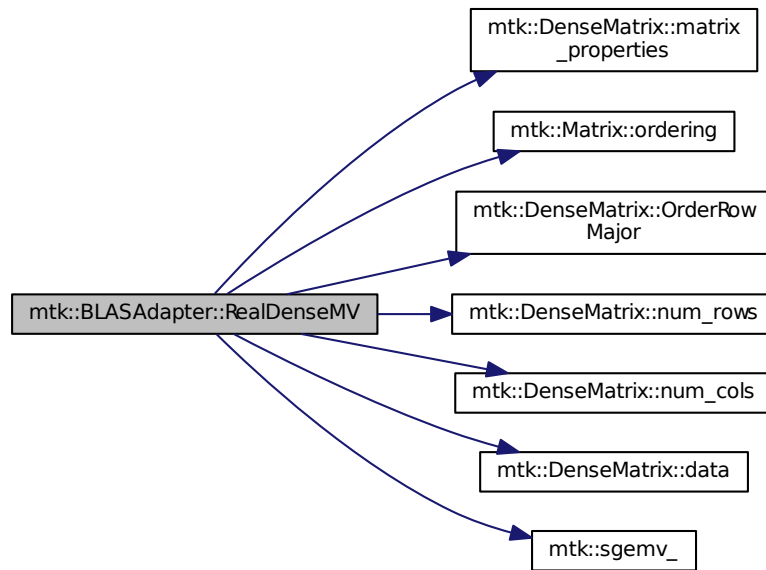
Here is the caller graph for this function:
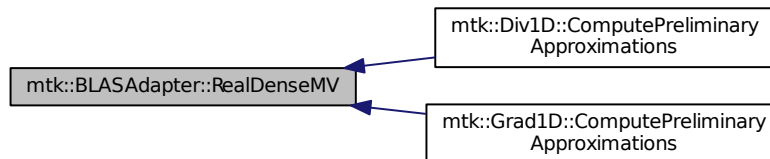
16.1.2.3   mtk::Real mtk::BLASAdapter::RealNRM2 (**Real** ∗ *in,* **int &** *in_length* **)** `[static]`

Parameters

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

Definition at line 276 of file mtk_blas_adapter.cc.

Here is the call graph for this function:

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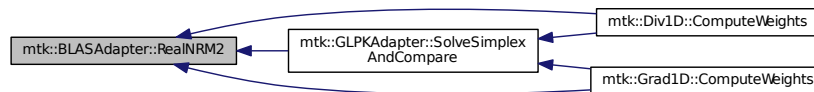
The documentation for this class was generated from the following files:

- include/mtk_blas_adapter.h

- src/mtk_blas_adapter.cc

16.2 mtk::DenseMatrix Class Reference

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

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

Collaboration diagram for mtk::DenseMatrix:
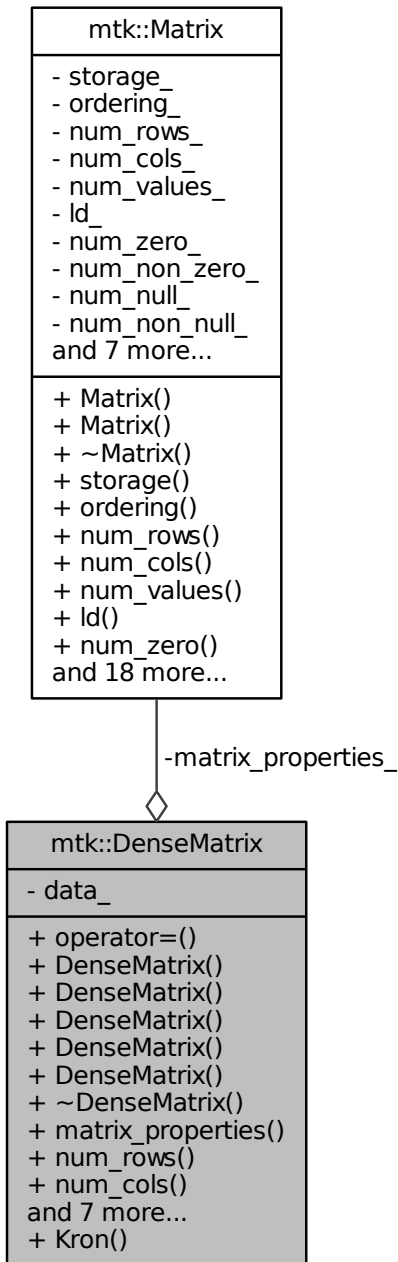
Public Member Functions

- DenseMatrix & operator= (const DenseMatrix &in)

Overloaded assignment operator.

- DenseMatrix ()

    Default constructor.

- DenseMatrix (const DenseMatrix &in)

    Copy constructor.

- DenseMatrix (const int &num_rows, const int &num_cols)

    Construct a dense matrix based on the given dimensions.

- DenseMatrix (const int &rank, const bool &padded, const bool &transpose)

    Construct a zero-rows-padded identity matrix.

- DenseMatrix (const Real ∗gen, const int &gen_length, const int &pro_length, const bool &transpose)

    Construct a dense Vandermonde matrix.

- ∼DenseMatrix ()

    Destructor.

- Matrix matrix_properties () const

    Provides access to the matrix data.

- int num_rows () const

    Gets the number of rows.

- int num_cols () const

    Gets the number of columns.

- Real ∗ data () const

    Provides access to the matrix value array.

- void SetOrdering (mtk::MatrixOrdering oo)

    Sets the ordering of the matrix.

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

    Gets a value on the given coordinates.

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

    Sets a value on the given coordinates.

- void Transpose ()

    Transpose this matrix.

- void OrderRowMajor ()

    Make the matrix row-wise ordered.

- void OrderColMajor ()

    Make the matrix column-wise ordered.

Static Public Member Functions

- static DenseMatrix Kron (const DenseMatrix &aa, const DenseMatrix &bb)

    Construct a dense matrix based on the Kronecker product of arguments.

Private Attributes

- Matrix matrix_properties_

    Data related to the matrix nature.

- Real ∗ data_

    Array holding the data in contiguouos position in memory.

Friends

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

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

16.2.1 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 98 of file mtk_dense_matrix.h.

16.2.2 Constructor & Destructor Documentation

16.2.2.1 mtk::DenseMatrix::DenseMatrix ( )

Definition at line 138 of file mtk_dense_matrix.cc.

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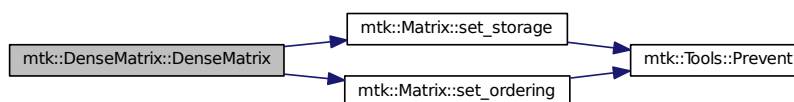
16.2.2.2 mtk::DenseMatrix::DenseMatrix (const DenseMatrix & *in*)

Parameters

`in`	*in*	Given matrix.

Definition at line 144 of file mtk_dense_matrix.cc.

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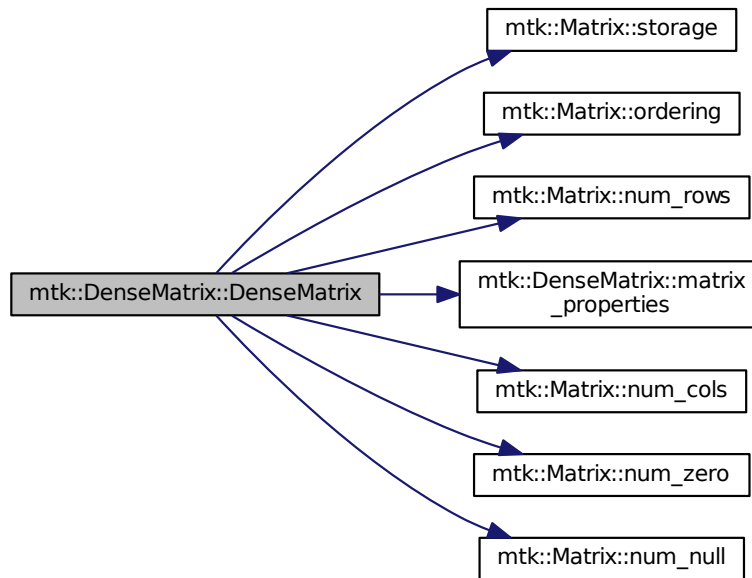
16.2.2.3 mtk::DenseMatrix::DenseMatrix (const int & *num_rows,* const int & *num_cols*)

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 177 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

16.2.2.4   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 199 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

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

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

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

This constructor generates a Vandermonde matrix, as defined above.

Obs∗∗. It 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 237 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

```
mtk::DenseMatrix::DenseMatrix ───▶ mtk::Tools::Prevent
```
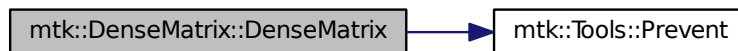
16.2.2.6   mtk::DenseMatrix::∼DenseMatrix (   )

Definition at line 285 of file mtk_dense_matrix.cc.

16.2.3   Member Function Documentation

16.2.3.1   mtk::Real ∗ mtk::DenseMatrix::data (   ) const

Returns

   Pointer to an array of mtk::Real.

Definition at line 316 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
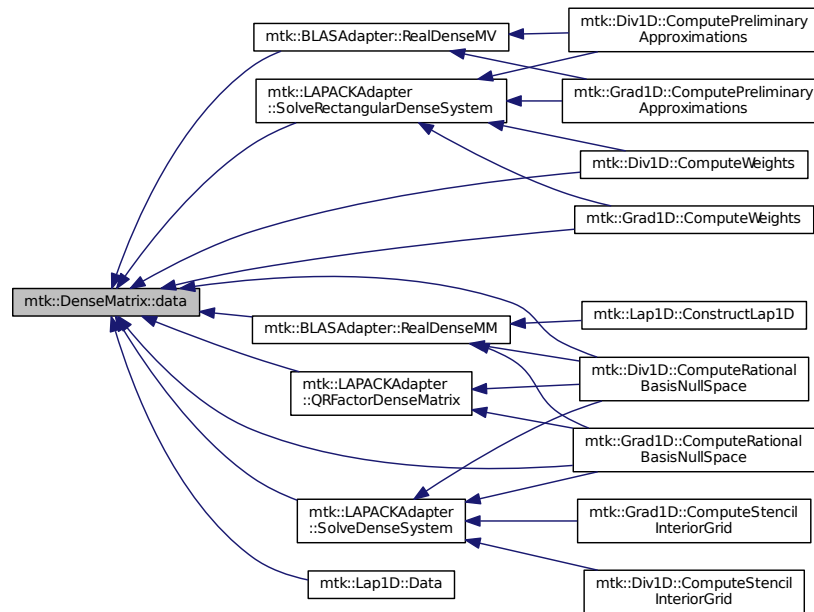
16.2.3.2   mtk::Real mtk::DenseMatrix::GetValue (const int & *row_coord,* const int & *col_coord*) const

Parameters

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

Returns

The required value at the specified coordinates.

Definition at line 321 of file mtk_dense_matrix.cc.

Here is the call graph for this function:

```
mtk::DenseMatrix::GetValue  ──────▶  mtk::Tools::Prevent
```

Here is the caller graph for this function:

```
                              mtk::Lap1D::ConstructLap1D
mtk::DenseMatrix::GetValue ◀
                              mtk::Lap1D::ReturnAsDense
                              Matrix
```
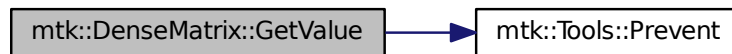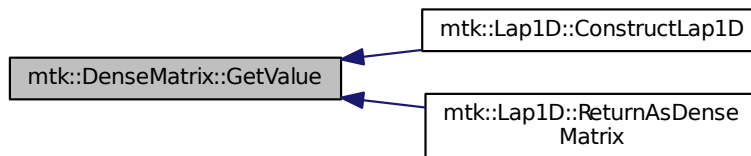
16.2.3.3 mtk::DenseMatrix mtk::DenseMatrix::Kron (const DenseMatrix & *aa,* const DenseMatrix & *bb*) [static]

Parameters

in	*aa*	First matrix.
in	*bb*	Second matrix.

Exceptions

std::bad_alloc	

Definition at line 463 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
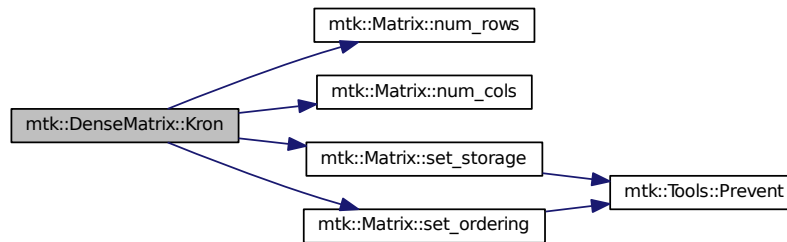
16.2.3.4 **mtk::Matrix** mtk::DenseMatrix::matrix_properties ( ) const

Returns

Pointer to a Matrix.

Definition at line 291 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
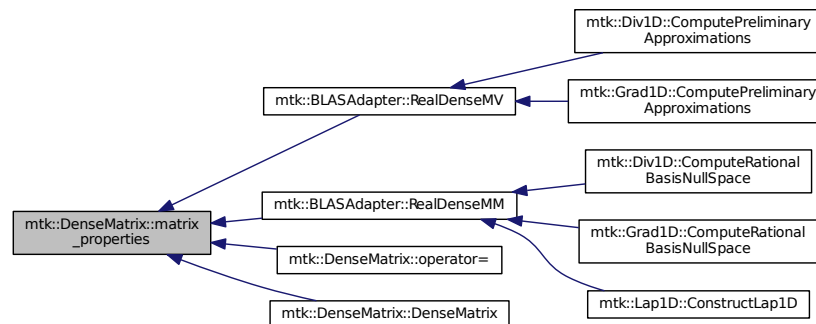
16.2.3.5 int mtk::DenseMatrix::num_cols ( ) const

Returns

Number of columns of the matrix.

Definition at line 311 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
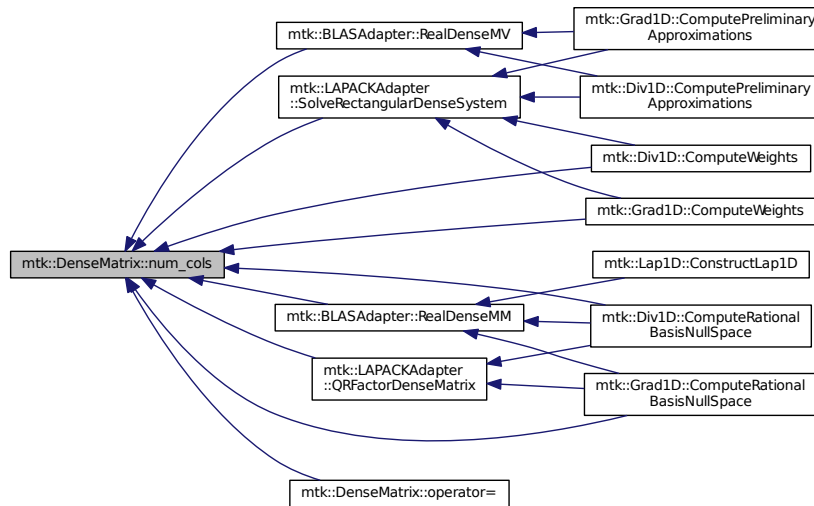
16.2.3.6   int mtk::DenseMatrix::num_rows (   ) const

Returns

Number of rows of the matrix.

Definition at line 306 of file mtk_dense_matrix.cc.

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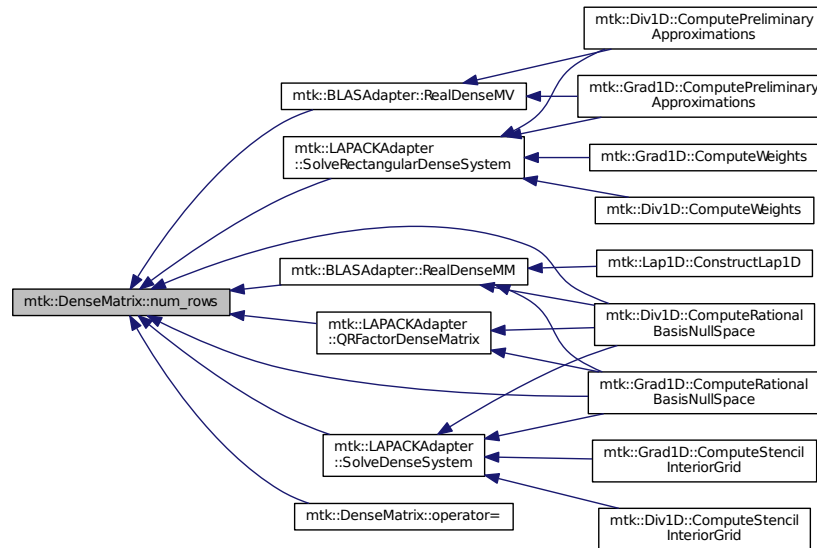
16.2.3.7 mtk::DenseMatrix & mtk::DenseMatrix::operator= (const DenseMatrix & *in*)

Definition at line 97 of file mtk_dense_matrix.cc.

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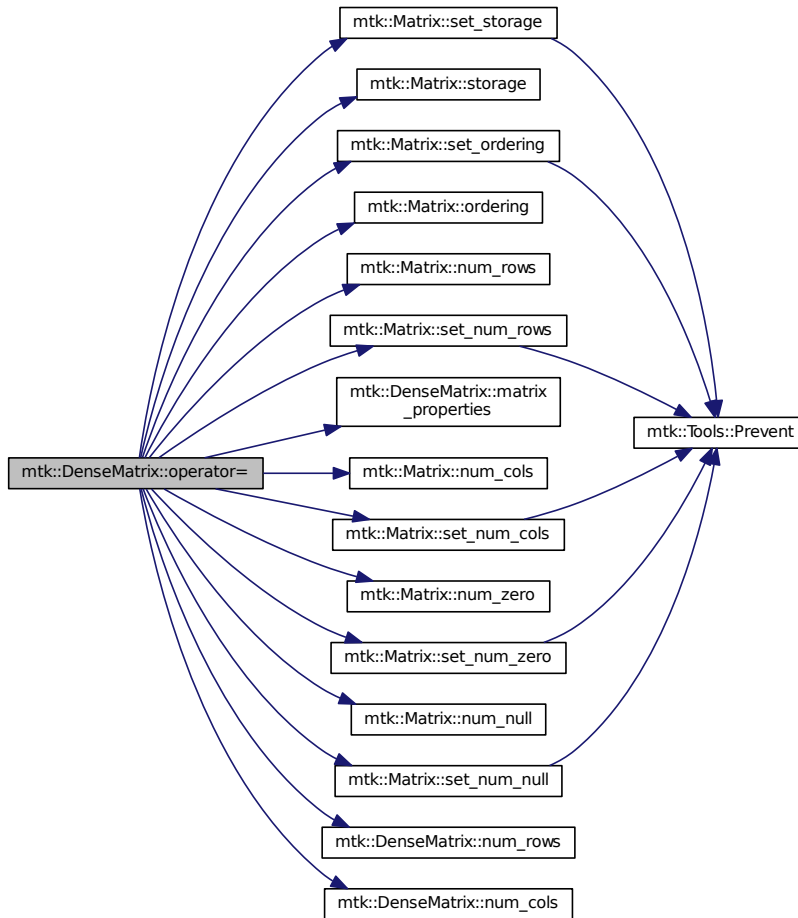
16.2.3.8   void mtk::DenseMatrix::OrderColMajor (   )

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

Definition at line 424 of file mtk_dense_matrix.cc.

16.2.3.9   void mtk::DenseMatrix::OrderRowMajor (   )

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

Definition at line 383 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
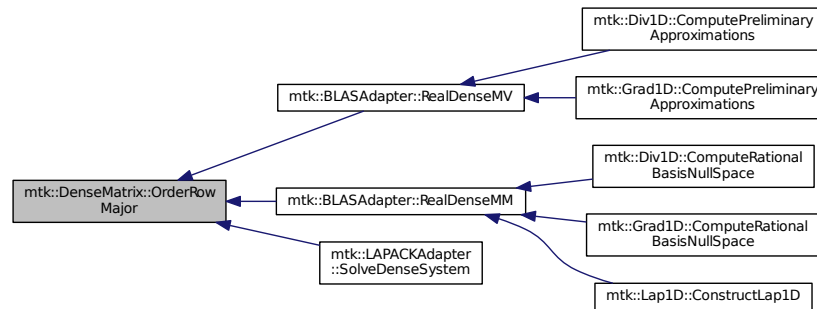
16.2.3.10   void mtk::DenseMatrix::SetOrdering (mtk::MatrixOrdering *oo*)

Parameters

`in`	*oo*	Ordering.

Returns

The required value at the specified coordinates.

Definition at line 296 of file mtk_dense_matrix.cc.

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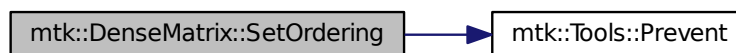
Here is the caller graph for this function:
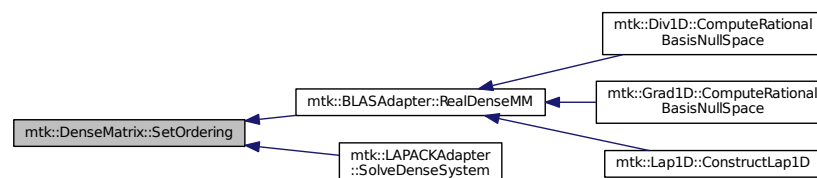
16.2.3.11 void mtk::DenseMatrix::SetValue (const int & *row_coord,* const int & *col_coord,* const **Real & *val*)**

Parameters

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

Definition at line 333 of file mtk_dense_matrix.cc.

Here is the call graph for this function:
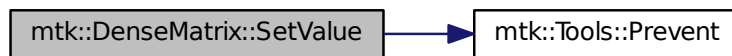
Here is the caller graph for this function:
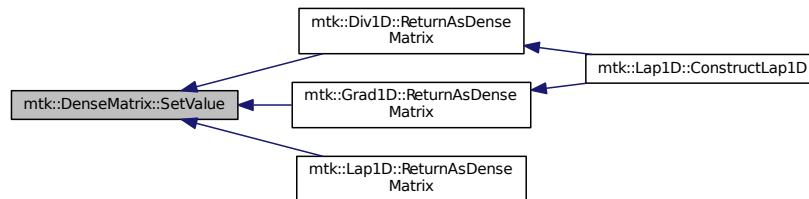
16.2.3.12 void mtk::DenseMatrix::Transpose (   )

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

Definition at line 346 of file mtk_dense_matrix.cc.

Here is the caller graph for this function:
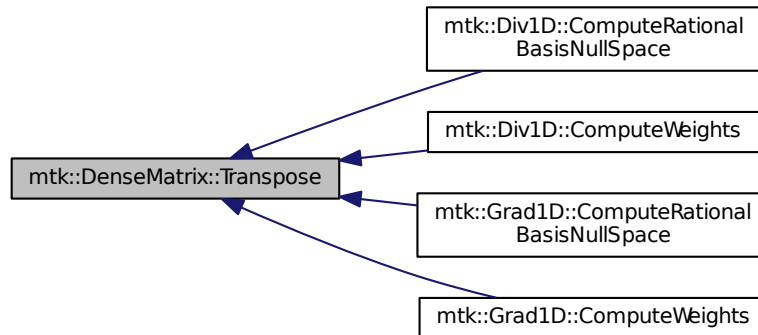
16.2.4 Friends And Related Function Documentation

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

Definition at line 75 of file mtk_dense_matrix.cc.

16.2.5 Member Data Documentation

16.2.5.1 Real∗ mtk::DenseMatrix::data_ `[private]`

Definition at line 274 of file mtk_dense_matrix.h.

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

Definition at line 272 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

16.3 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=()           │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + DenseMatrix()         │
                    │ + ~DenseMatrix()        │
                    │ + matrix_properties()   │
                    │ + num_rows()            │
                    │ + num_cols()            │
                    │ and 7 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()      │
                    │ + weights_crs()         │
                    │ + weights_cbs()         │
                    │ + 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.

- Real ∗ weights_crs (void)

    Return collection of weights as computed by the CRSA.

- Real ∗ weights_cbs (void)

    Return collection of weights as computed by the CBSA.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid)

    Return the operator as a dense matrix.

Private Member Functions

- bool ComputeStencilInteriorGrid (void)

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

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

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

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

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

    Stage 3 of the CBS Algorithm.

Private Attributes

- int order_accuracy_

    Order of numerical accuracy of the operator.
- int dim_null_

    Dim. null-space for boundary approximations.
- int num_bndy_coeffs_

    Req. coeffs. per bndy pt. uni. order accuracy.
- int divergence_length_

    Length of the output array.
- int minrow_

    Row from the optimizer with the minimum rel. nor.
- int row_

    Row currently processed by the optimizer.
- mtk::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.

16.3.1 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.

16.3.2 Constructor & Destructor Documentation

16.3.2.1 mtk::Div1D::Div1D ( )

Definition at line 125 of file mtk_div_1d.cc.

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

Parameters

in	*div*	Given divergence.

Definition at line 140 of file mtk_div_1d.cc.

16.3.2.3 mtk::Div1D::∼Div1D ( )

Definition at line 155 of file mtk_div_1d.cc.

16.3.3 Member Function Documentation

16.3.3.1 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 1306 of file mtk_div_1d.cc.

16.3.3.2 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 662 of file mtk_div_1d.cc.

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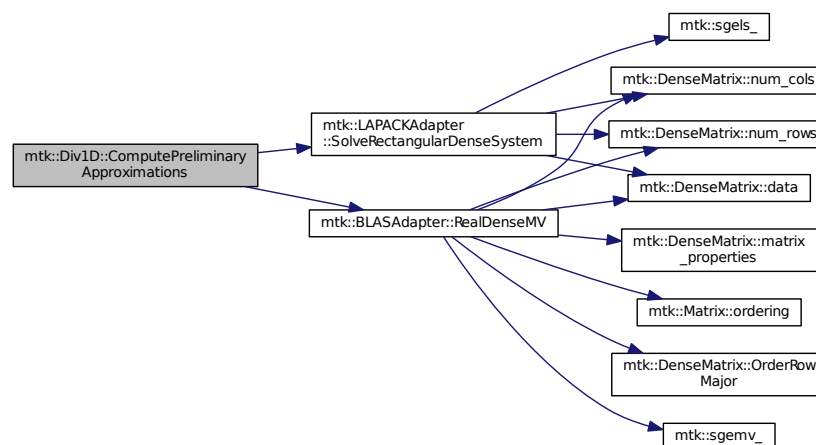
16.3.3.3 bool mtk::Div1D::ComputeRationalBasisNullSpace (void) `[private]`

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 486 of file mtk_div_1d.cc.

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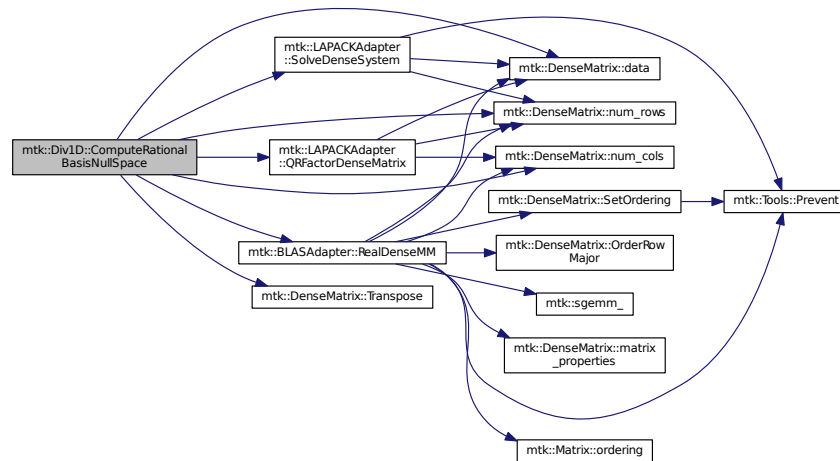
16.3.3.4 bool mtk::Div1D::ComputeStencilBoundaryGrid (void) `[private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.

2. Compute alpha values.

3. Compute the mimetic boundary approximations.

Definition at line 1207 of file mtk_div_1d.cc.

16.3.3.5 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 387 of file mtk_div_1d.cc.

Here is the call graph for this function:

```
mtk::Div1D::ComputeStencil        mtk::LAPACKAdapter          mtk::Tools::Prevent
InteriorGrid               -->    ::SolveDenseSystem    -->   mtk::DenseMatrix::num_rows
                                                             mtk::DenseMatrix::data
```
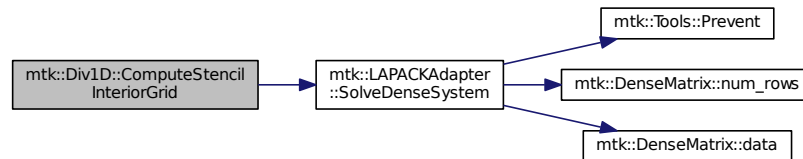
16.3.3.6   **bool mtk::Div1D::ComputeWeights (void)** `[private]`

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 882 of file mtk_div_1d.cc.

Here is the call graph for this function:

```
mtk::Div1D::ComputeWeights  -->  mtk::DenseMatrix::data
                                 mtk::DenseMatrix::num_cols
                                 mtk::LAPACKAdapter::SolveRectangularDenseSystem  -->  mtk::sgels_
                                                                                      mtk::DenseMatrix::num_rows
                                 mtk::DenseMatrix::Transpose
                                 mtk::GLPKAdapter::SolveSimplexAndCompare  -->  mtk::BLASAdapter::RealNRM2  -->  mtk::snrm2_
```
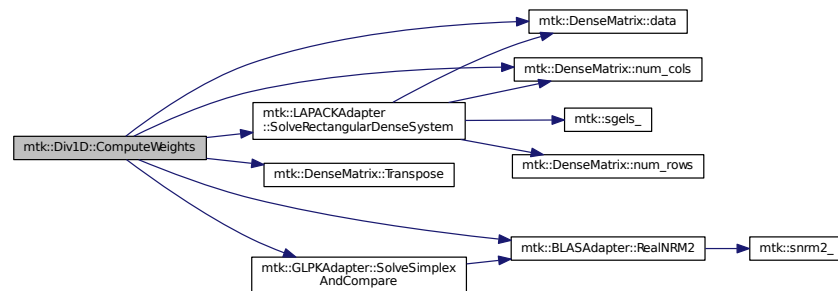
16.3.3.7 bool mtk::Div1D::ConstructDiv1D (int *order_accuracy* = kDefaultOrderAccuracy, mtk::Real *mimetic_threshold* = kDefaultMimeticThreshold)

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:

```
mtk::Div1D::ConstructDiv1D  ──▶  mtk::Tools::Prevent
```

Here is the caller graph for this function:

```
mtk::Div1D::ConstructDiv1D  ◀──  mtk::Lap1D::ConstructLap1D
```

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

Returns

The operator as a dense matrix.

1. Insert mimetic boundary at the west.

2. Insert coefficients for the interior of the grid.

3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 325 of file mtk_div_1d.cc.

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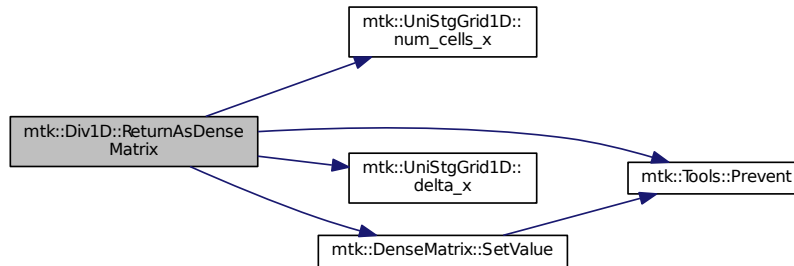
Here is the caller graph for this function:

16.3.3.9 mtk::Real ∗ mtk::Div1D::weights_cbs (void)

Returns

Collection of weights as computed by the CBSA.

Definition at line 320 of file mtk_div_1d.cc.

16.3.3.10 mtk::Real ∗ mtk::Div1D::weights_crs (void)

Returns

Collection of weights as computed by the CRSA.

Definition at line 315 of file mtk_div_1d.cc.

16.3.4 Friends And Related Function Documentation

16.3.4.1 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.

16.3.5 Member Data Documentation

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

Definition at line 182 of file mtk_div_1d.h.

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

Definition at line 173 of file mtk_div_1d.h.

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

Definition at line 187 of file mtk_div_1d.h.

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

Definition at line 175 of file mtk_div_1d.h.

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

Definition at line 186 of file mtk_div_1d.h.

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

Definition at line 189 of file mtk_div_1d.h.

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

Definition at line 177 of file mtk_div_1d.h.

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

Definition at line 174 of file mtk_div_1d.h.

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

Definition at line 172 of file mtk_div_1d.h.

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

Definition at line 183 of file mtk_div_1d.h.

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

Definition at line 180 of file mtk_div_1d.h.

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

Definition at line 178 of file mtk_div_1d.h.

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

Definition at line 185 of file mtk_div_1d.h.

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

Definition at line 184 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

16.4 mtk::GLPKAdapter Class Reference

Adapter class for the GLPK API.

`#include <mtk_glpk_adapter.h>`

Collaboration diagram for mtk::GLPKAdapter:

```
+---------------------------+
|     mtk::GLPKAdapter      |
+---------------------------+
|                           |
+---------------------------+
| + SolveSimplexAndCompare()|
+---------------------------+
```
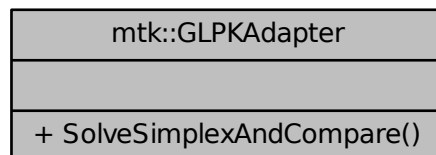
Static Public Member Functions

- 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.

16.4.1 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 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.

16.4.2 Member Function Documentation

16.4.2.1 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.

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 78 of file mtk_glpk_adapter.cc.

Here is the call graph for this function:


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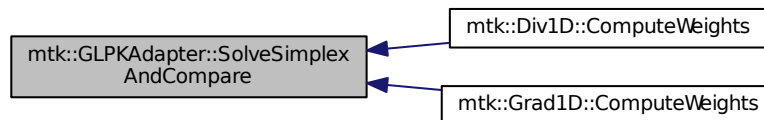
The documentation for this class was generated from the following files:

- include/mtk_glpk_adapter.h
- src/mtk_glpk_adapter.cc

16.5 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=()           │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + DenseMatrix()         │
│ + ~DenseMatrix()        │
│ + matrix_properties()   │
│ + num_rows()            │
│ + num_cols()            │
│ and 7 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()     │
│ + weights_crs()         │
│ + weights_cbs()         │
│ + ReturnAsDenseMatrix() │
│ - 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.

- Real ∗ weights_crs (void)

    Return collection of weights as computed by the CRSA.

- Real ∗ weights_cbs (void)

    Return collection of weights as computed by the CBSA.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid)

    Return the operator as a dense matrix.

Private Member Functions

- bool ComputeStencilInteriorGrid (void)

    Stage 1 of the CBS Algorithm.

- bool ComputeRationalBasisNullSpace (void)

    Stage 2.1 of the CBS Algorithm.

- bool ComputePreliminaryApproximations (void)

    Stage 2.2 of the CBS Algorithm.

- bool ComputeWeights (void)

    Stage 2.3 of the CBS Algorithm.

- bool ComputeStencilBoundaryGrid (void)

    Stage 2.4 of the CBS Algorithm.

- bool AssembleOperator (void)

    Stage 3 of the CBS Algorithm.

Private Attributes

- int order_accuracy_

    Order of numerical accuracy of the operator.

- int dim_null_

    Dim. null-space for boundary approximations.

- int num_bndy_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.

- mtk::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.

16.5.1   Detailed Description

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

Definition at line 81 of file mtk_grad_1d.h.

16.5.2   Constructor & Destructor Documentation

16.5.2.1   mtk::Grad1D::Grad1D (   )

Definition at line 129 of file mtk_grad_1d.cc.

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

Parameters

in	*div*	Given divergence.

Definition at line 145 of file mtk_grad_1d.cc.

16.5.2.3   mtk::Grad1D::∼Grad1D (   )

Definition at line 161 of file mtk_grad_1d.cc.

16.5.3 Member Function Documentation

16.5.3.1 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 1344 of file mtk_grad_1d.cc.

16.5.3.2 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 680 of file mtk_grad_1d.cc.

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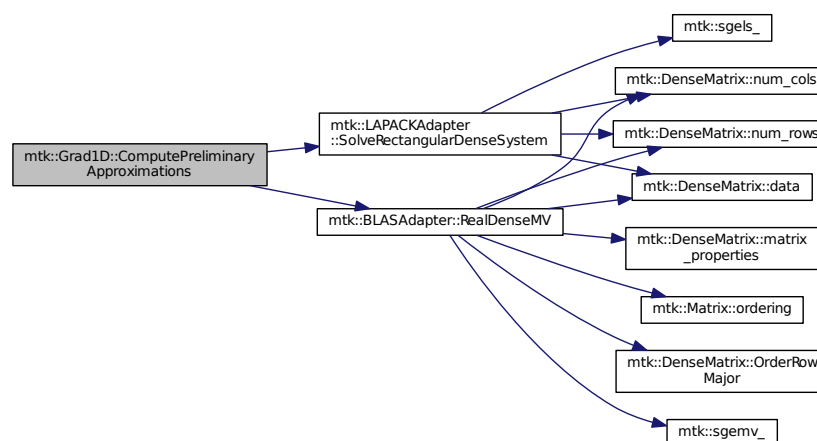
16.5.3.3 bool mtk::Grad1D::ComputeRationalBasisNullSpace (void) `[private]`

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 497 of file mtk_grad_1d.cc.

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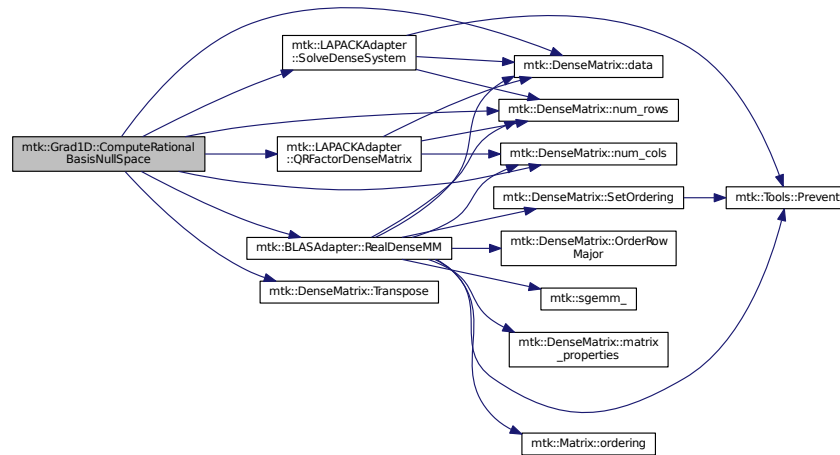
16.5.3.4 bool mtk::Grad1D::ComputeStencilBoundaryGrid (void) `[private]`

Compute mimetic stencil approximating at boundary.

1. Collect lambda values.

2. Compute alpha values.

3. Compute the mimetic boundary approximations.

Definition at line 1238 of file mtk_grad_1d.cc.

16.5.3.5 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 401 of file mtk_grad_1d.cc.

Here is the call graph for this function:

16.5.3.6 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 900 of file mtk_grad_1d.cc.

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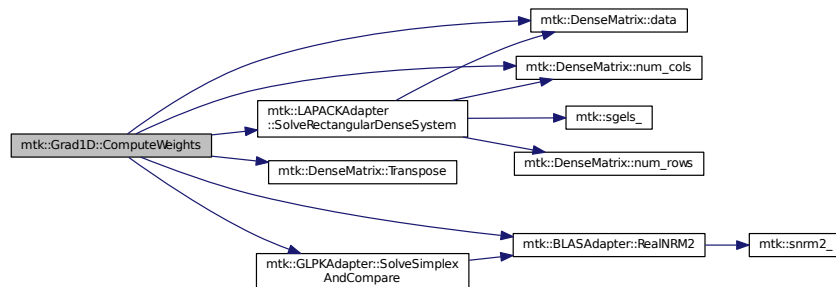
16.5.3.7 **bool mtk::Grad1D::ConstructGrad1D (** int *order_accuracy* **= kDefaultOrderAccuracy,** **Real** *mimetic_threshold* **= kDefaultMimeticThreshold)**

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:
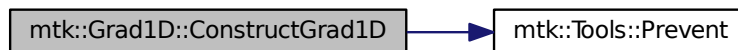
Here is the caller graph for this function:

16.5.3.8 **mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix (** const **UniStgGrid1D &** *grid* **)**

Returns

The operator as a dense matrix.

1. Insert mimetic boundary at the west.

2. Insert coefficients for the interior of the grid.

3. Impose center-skew symmetry by permuting the mimetic boundaries.

Definition at line 335 of file mtk_grad_1d.cc.

Here is the call graph for this function:
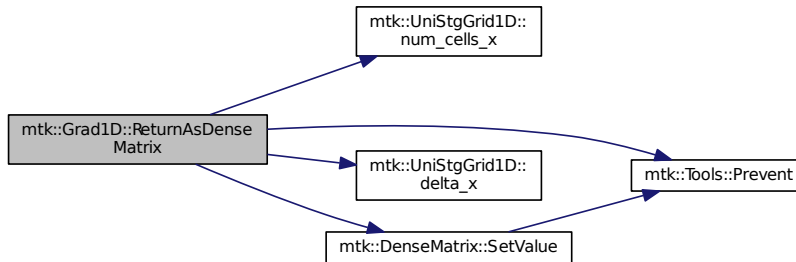
Here is the caller graph for this function:
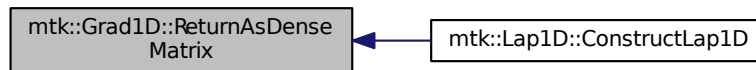
16.5.3.9   mtk::Real ∗ **mtk::Grad1D::weights_cbs (void)**

Returns

  Collection of weights as computed by the CBSA.

Definition at line 330 of file mtk_grad_1d.cc.

16.5.3.10   mtk::Real ∗ **mtk::Grad1D::weights_crs (void)**

Returns

  Success of the solution.

Definition at line 325 of file mtk_grad_1d.cc.

16.5.4   Friends And Related Function Documentation

16.5.4.1   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.

16.5.5 Member Data Documentation

16.5.5.1 Real∗ mtk::Grad1D::coeffs_interior_ `[private]`

Definition at line 183 of file mtk_grad_1d.h.

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

Definition at line 173 of file mtk_grad_1d.h.

16.5.5.3 Real∗ mtk::Grad1D::gradient_ `[private]`

Definition at line 188 of file mtk_grad_1d.h.

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

Definition at line 176 of file mtk_grad_1d.h.

16.5.5.5 Real∗ mtk::Grad1D::mim_bndy_ `[private]`

Definition at line 187 of file mtk_grad_1d.h.

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

Definition at line 190 of file mtk_grad_1d.h.

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

Definition at line 178 of file mtk_grad_1d.h.

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

Definition at line 174 of file mtk_grad_1d.h.

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

Definition at line 175 of file mtk_grad_1d.h.

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

Definition at line 172 of file mtk_grad_1d.h.

16.5.5.11 Real∗ mtk::Grad1D::prem_apps_ `[private]`

Definition at line 184 of file mtk_grad_1d.h.

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

Definition at line 181 of file mtk_grad_1d.h.

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

Definition at line 179 of file mtk_grad_1d.h.

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

Definition at line 186 of file mtk_grad_1d.h.

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

Definition at line 185 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

16.6 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_               │
│ - mimetic_threshold_        │
├─────────────────────────────┤
│ + Lap1D()                   │
│ + Lap1D()                   │
│ + ~Lap1D()                  │
│ + ConstructLap1D()          │
│ + ReturnAsDenseMatrix()     │
│ + Data()                    │
└─────────────────────────────┘
```
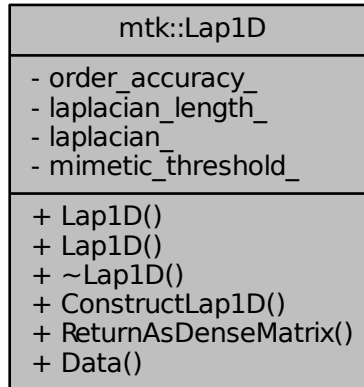
Public Member Functions

- Lap1D ()

  Default constructor.

- Lap1D (const Lap1D &lap)

  Copy constructor.

- ~Lap1D ()

  Destructor.

- bool ConstructLap1D (int order_accuracy=kDefaultOrderAccuracy, Real mimetic_threshold=kDefaultMimetic-Threshold)

  Factory method implementing the CBS Algorithm to build operator.

- DenseMatrix ReturnAsDenseMatrix (const UniStgGrid1D &grid)

  Return the operator as a dense matrix.

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

  Return the operator as a dense array.

Private Attributes

- int order_accuracy_

  Order of numerical accuracy of the operator.

- int laplacian_length_

  Length of the output array.

- Real ∗ laplacian_

  Output array containing the operator and weights.

- Real mimetic_threshold_

  < *Mimetic threshold.*

Friends

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

    Output stream operator for printing.

16.6.1 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.

16.6.2 Constructor & Destructor Documentation

16.6.2.1 mtk::Lap1D::Lap1D ( )

Definition at line 108 of file mtk_lap_1d.cc.

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

Parameters

in	*lap*	Given Laplacian.

16.6.2.3 mtk::Lap1D::∼Lap1D ( )

Definition at line 113 of file mtk_lap_1d.cc.

16.6.3 Member Function Documentation

16.6.3.1 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.

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

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

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

Definition at line 119 of file mtk_lap_1d.cc.

Here is the call graph for this function:
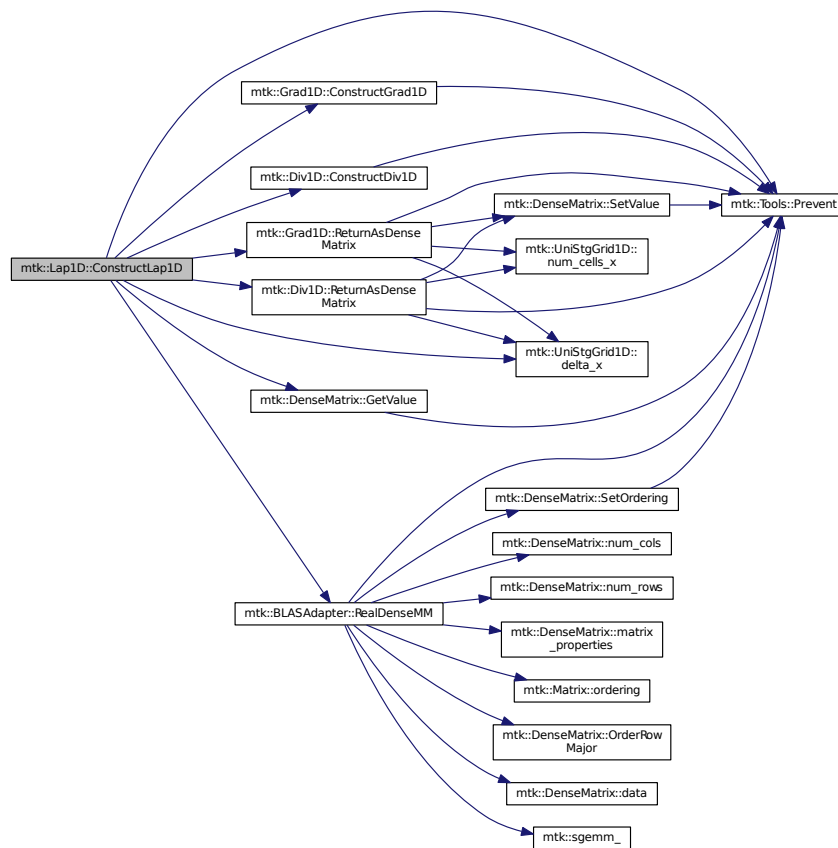
16.6.3.2 **mtk::Real** ∗ **mtk::Lap1D::Data (const UniStgGrid1D &** *grid* **)**

Returns

The operator as a dense array.

Definition at line 332 of file mtk_lap_1d.cc.

Here is the call graph for this function:

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

Returns

The operator as a dense matrix.

1. Extract mimetic coefficients from the west boundary.

2. Extract interior coefficients.

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

Note

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

Definition at line 265 of file mtk_lap_1d.cc.

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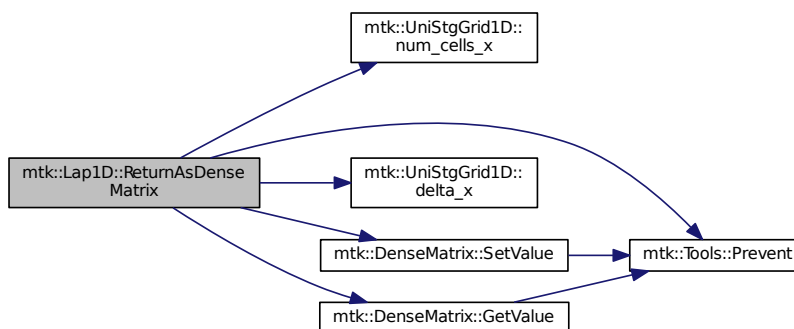
16.6.4 Friends And Related Function Documentation

16.6.4.1 std::ostream& operator$\ll$ **(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.

16.6.5 Member Data Documentation

16.6.5.1 Real$*$ **mtk::Lap1D::laplacian_** `[private]`

Definition at line 120 of file mtk_lap_1d.h.

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

Definition at line 118 of file mtk_lap_1d.h.

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

Definition at line 122 of file mtk_lap_1d.h.

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

Definition at line 117 of file mtk_lap_1d.h.

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

- include/mtk_lap_1d.h
- src/mtk_lap_1d.cc

16.7 mtk::LAPACKAdapter Class Reference

Adapter class for the LAPACK API.

`#include <mtk_lapack_adapter.h>`

Collaboration diagram for mtk::LAPACKAdapter:

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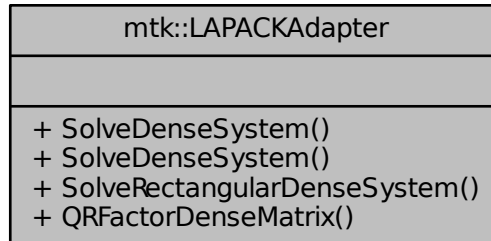
Static Public Member Functions

- 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 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.

16.7.1 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 the numerical methods implemented in the LAPACK.

The **LAPACK** 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 89 of file mtk_lapack_adapter.h.

16.7.2 Member Function Documentation

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

Adapts the MTK to LAPACK's routine.

Parameters

`in,out`	*matrix*	Input matrix.

Returns

> Matrix $\mathbf{Q}$.

Exceptions

std::bad_alloc	

Definition at line 515 of file mtk_lapack_adapter.cc.

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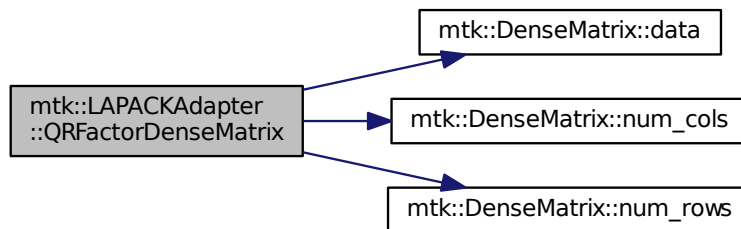
Here is the caller graph for this function:
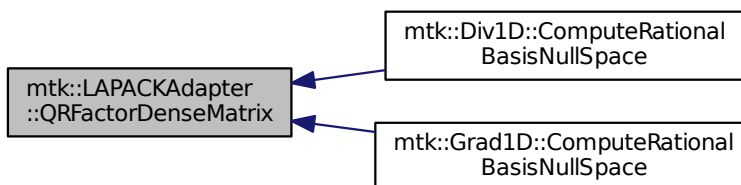
16.7.2.2   int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & *mm,* **mtk::Real** $*$ *rhs* **)** `[static]`

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 427 of file mtk_lapack_adapter.cc.

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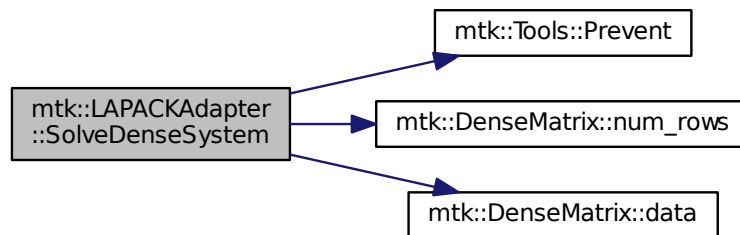
Here is the caller graph for this function:
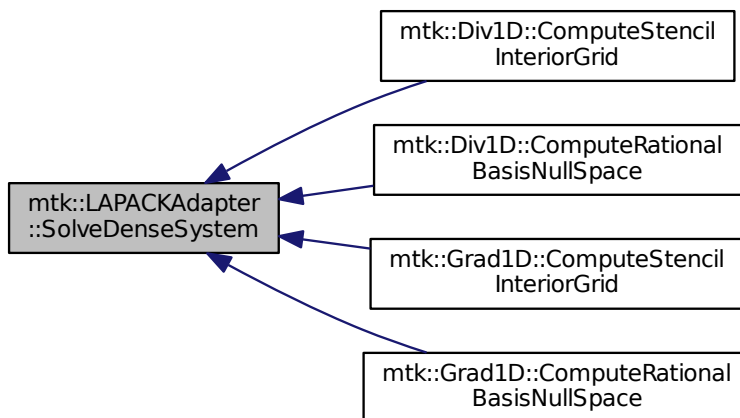
16.7.2.3   int mtk::LAPACKAdapter::SolveDenseSystem (mtk::DenseMatrix & *mm,* mtk::DenseMatrix & *rr*) `[static]`

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 463 of file mtk_lapack_adapter.cc.

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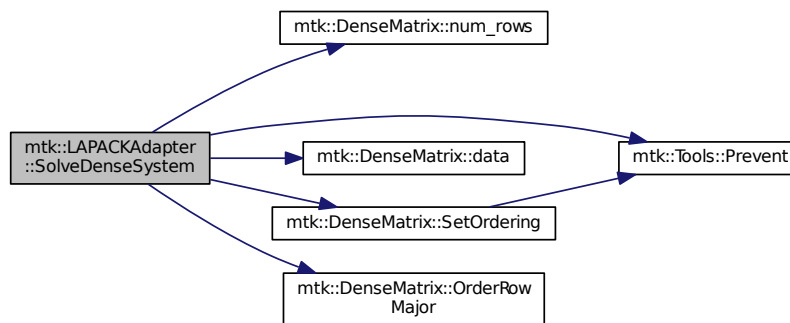
16.7.2.4 int mtk::LAPACKAdapter::SolveRectangularDenseSystem (const mtk::DenseMatrix & *aa,* mtk::Real ∗ *ob_,* int *ob_ld_*
) `[static]`

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 716 of file mtk_lapack_adapter.cc.
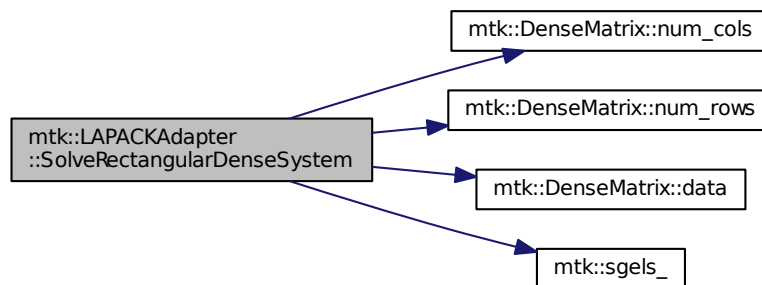
Here is the call graph for this function:

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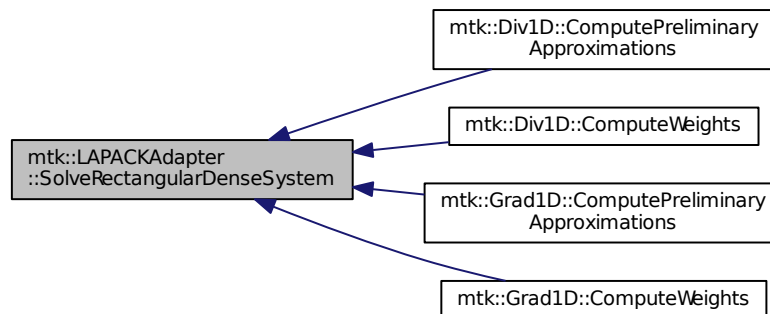
The documentation for this class was generated from the following files:

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

16.8   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 ()

    Destructor.

- MatrixStorage storage () const

    Gets the type of storage of this matrix.

- MatrixOrdering ordering () const

    Gets the ordering of this matrix.

- int num_rows () const

    Gets the number of rows.

- int num_cols () const

    Gets the number of rows.

- int num_values () const

    Gets the number of values.

- int ld () const

    Gets the matrix' leading dimension.

- int num_zero () const

    Gets the number of zeros.

- int num_non_zero () const

    Gets the number of non-zero values.

- int num_null () const

    Gets the number of null values.

- int num_non_null () const

    Gets the number of non-null values.

- int kl () const

    Gets the number of lower diagonals.

- int ku () const

    Gets the number of upper diagonals.

- int bandwidth () const

    Gets the bandwidth.

- Real abs_density () const

    Gets the absolute density.

- Real rel_density () const

    Gets the relative density.

- Real abs_sparsity () const

    Gets the Absolute sparsity.

- Real rel_sparsity () const

    Gets the Relative sparsity.

- void set_storage (const MatrixStorage &tt)

    Sets the storage type of the matrix.

- void set_ordering (const MatrixOrdering &oo)

    Sets the ordering of the matrix.

- void set_num_rows (int num_rows)

    Sets the number of rows of the matrix.

- void set_num_cols (int num_cols)

    Sets the number of columns of the matrix.

- void set_num_zero (int in)

    Sets the number of zero values of the matrix that matter.

- void set_num_null (int in)

    Sets the number of zero values of the matrix that DO NOT matter.

- void IncreaseNumZero ()

    Increases the number of values that equal zero but with meaning.

- void IncreaseNumNull ()

    Increases the number of values that equal zero but with no meaning.

Private Attributes

- MatrixStorage storage_

    What type of matrix is this?
- MatrixOrdering ordering_

    What kind of ordering is it following?
- int num_rows_

    Number of rows.
- int num_cols_

    Number of columns.
- int num_values_

    Number of total values in matrix.
- int ld_

    Elements between successive rows when row-major.
- int num_zero_

    Number of zeros.
- int num_non_zero_

    Number of non-zero values.
- int num_null_

    Number of null (insignificant) values.
- int num_non_null_

    Number of null (significant) values.
- int kl_

    Number of lower diagonals on a banded matrix.
- int ku_

    Number of upper diagonals on a banded matrix.
- int bandwidth_

    Bandwidth of the matrix.
- Real abs_density_

    Absolute density of matrix.
- Real rel_density_

    Relative density of matrix.
- Real abs_sparsity_

    Absolute sparsity of matrix.
- Real rel_sparsity_

    Relative sparsity of matrix.

16.8.1 Detailed Description

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

Definition at line 75 of file mtk_matrix.h.

16.8.2 Constructor & Destructor Documentation

16.8.2.1 mtk::Matrix::Matrix ( )

Definition at line 72 of file mtk_matrix.cc.

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

Parameters

in		*in*	Given matrix.

Definition at line 91 of file mtk_matrix.cc.

16.8.2.3 mtk::Matrix::∼Matrix ( )

Definition at line 110 of file mtk_matrix.cc.

16.8.3 Member Function Documentation

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

See Also

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

Returns

Absolute density of the matrix.

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

See Also

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

Returns

Absolute sparsity of the matrix.

Definition at line 182 of file mtk_matrix.cc.

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

Returns

Bandwidth of the matrix.

Definition at line 172 of file mtk_matrix.cc.

16.8.3.4 void mtk::Matrix::IncreaseNumNull ( )

Todo Review the definition of sparse matrices properties.

Definition at line 279 of file mtk_matrix.cc.

16.8.3.5 void mtk::Matrix::IncreaseNumZero (   )

Todo Review the definition of sparse matrices properties.

Definition at line 269 of file mtk_matrix.cc.

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

Returns

Number of lower diagonals.

Definition at line 162 of file mtk_matrix.cc.

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

Returns

Number of upper diagonals.

Definition at line 167 of file mtk_matrix.cc.

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

Leading dimension of the data array is the number of elements between successive rows (for row major storage) in memory. Most of the cases, the leading dimension is the same as the number of columns.

Returns

Leading dimension of the matrix.

Definition at line 137 of file mtk_matrix.cc.

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

Returns

Number of rows of the matrix.

Definition at line 127 of file mtk_matrix.cc.

Here is the caller graph for this function:
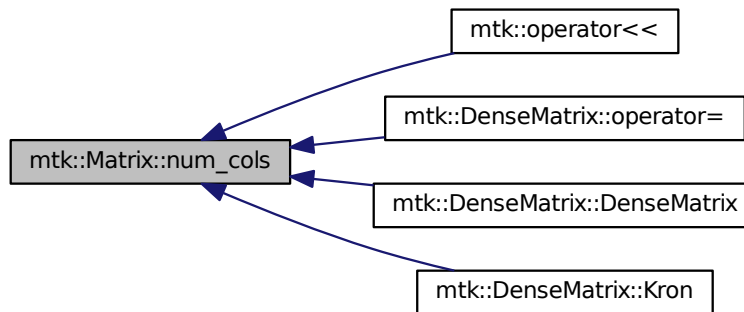
16.8.3.10   int mtk::Matrix::num_non_null (   ) const

See Also

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

Returns

Number of non-null values of the matrix.

Definition at line 157 of file mtk_matrix.cc.

16.8.3.11   int mtk::Matrix::num_non_zero (   ) const

Returns

Number of non-zero values of the matrix.

Definition at line 147 of file mtk_matrix.cc.

16.8.3.12   int mtk::Matrix::num_null (   ) const

See Also

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

Returns

Number of null values of the matrix.

Definition at line 152 of file mtk_matrix.cc.

Here is the caller graph for this function:

```
mtk::Matrix::num_null  ←  mtk::DenseMatrix::operator=
                       ←  mtk::DenseMatrix::DenseMatrix
```
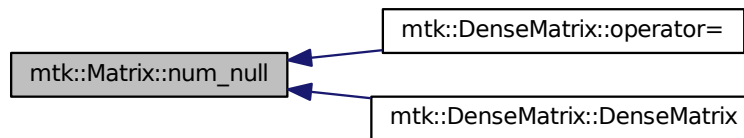
16.8.3.13   int mtk::Matrix::num_rows ( ) const

Returns

Number of rows of the matrix.

Definition at line 122 of file mtk_matrix.cc.

Here is the caller graph for this function:

```
mtk::Matrix::num_rows  ←  mtk::operator<<
                       ←  mtk::DenseMatrix::operator=
                       ←  mtk::DenseMatrix::DenseMatrix
                       ←  mtk::DenseMatrix::Kron
```
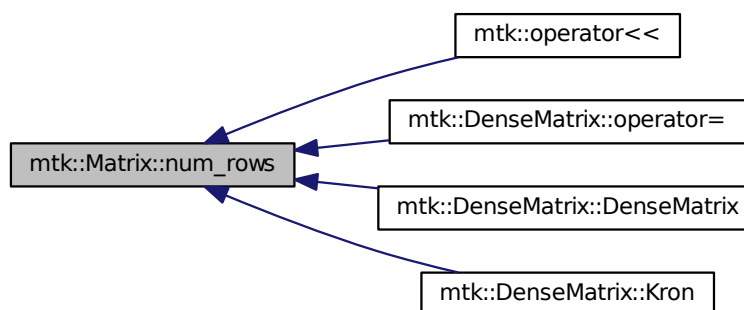
16.8.3.14   int mtk::Matrix::num_values ( ) const

Returns

Number of values of the matrix.

Definition at line 132 of file mtk_matrix.cc.

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

Returns

Number of zeros of the matrix.

Definition at line 142 of file mtk_matrix.cc.

Here is the caller graph for this function:
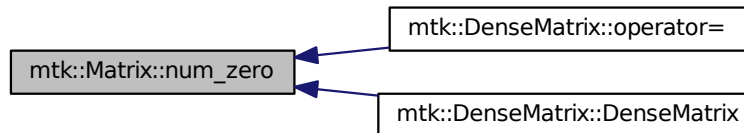
16.8.3.16 mtk::MatrixOrdering mtk::Matrix::ordering ( ) const

Definition at line 117 of file mtk_matrix.cc.

Here is the caller graph for this function:
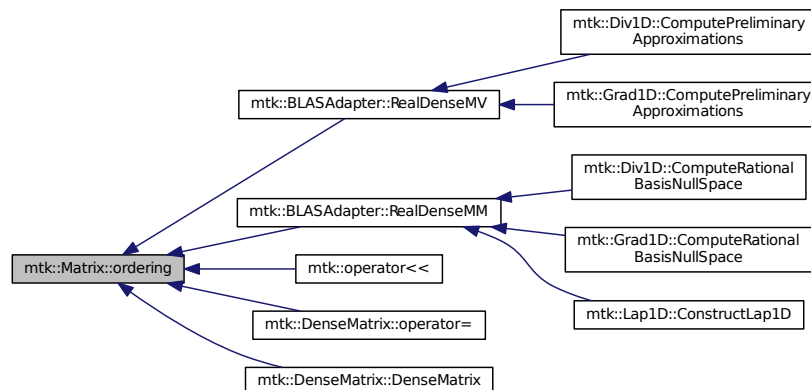
16.8.3.17 mtk::Real mtk::Matrix::rel_density ( ) const

See Also

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

Returns

Relative density of the matrix.

Definition at line 177 of file mtk_matrix.cc.

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

See Also

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

Returns

Relative sparsity of the matrix.

Definition at line 187 of file mtk_matrix.cc.

16.8.3.19 void mtk::Matrix::set_num_cols (int *num_cols*)

Parameters

in	*num_cols*	Number of columns.

Definition at line 229 of file mtk_matrix.cc.

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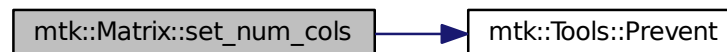
Here is the caller graph for this function:

16.8.3.20 void mtk::Matrix::set_num_null (int *in*)

Parameters

in		*in*	Number of zero values.

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

Definition at line 255 of file mtk_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

16.8.3.21   void mtk::Matrix::set_num_rows (int *num_rows*)

Parameters

in		*num_rows*	Number of rows.

Definition at line 217 of file mtk_matrix.cc.

Here is the call graph for this function:

Here is the caller graph for this function:

| mtk::Matrix::set_num_rows | ◄── | mtk::DenseMatrix::operator= |

16.8.3.22   void mtk::Matrix::set_num_zero (int *in*)

Parameters

in		*in*	Number of zero values.

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

Definition at line 241 of file mtk_matrix.cc.

Here is the call graph for this function:

| mtk::Matrix::set_num_zero | ──► | mtk::Tools::Prevent |

Here is the caller graph for this function:

| mtk::Matrix::set_num_zero | ◄── | mtk::DenseMatrix::operator= |
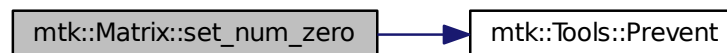
16.8.3.23   void mtk::Matrix::set_ordering (const MatrixOrdering & *oo*)

See Also

MatrixOrdering

Parameters

in		*oo*	Ordering of the matrix.

Definition at line 204 of file mtk_matrix.cc.

Here is the call graph for this function:

```
mtk::Matrix::set_ordering  →  mtk::Tools::Prevent
```

Here is the caller graph for this function:

```
                              mtk::DenseMatrix::operator=
mtk::Matrix::set_ordering  ←  mtk::DenseMatrix::DenseMatrix
                              mtk::DenseMatrix::Kron
```
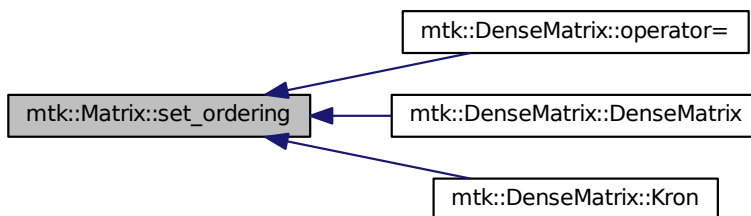
16.8.3.24   void mtk::Matrix::set_storage (const MatrixStorage & *tt*)

See Also

> [MatrixStorage](#)

Parameters

in		*tt*	Type of the matrix storage.

Definition at line 192 of file mtk_matrix.cc.

Here is the call graph for this function:

```
┌────────────────────────┐        ┌──────────────────────┐
│ mtk::Matrix::set_storage │ ─────> │ mtk::Tools::Prevent  │
└────────────────────────┘        └──────────────────────┘
```

Here is the caller graph for this function:
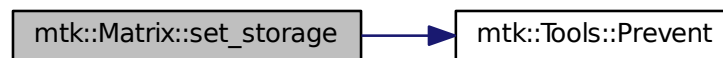
```
                                ┌──────────────────────────────┐
                                │ mtk::DenseMatrix::operator=    │
                                └──────────────────────────────┘
┌────────────────────────┐      ┌──────────────────────────────────┐
│ mtk::Matrix::set_storage │ <─── │ mtk::DenseMatrix::DenseMatrix     │
└────────────────────────┘      └──────────────────────────────────┘
                                ┌──────────────────────────────┐
                                │ mtk::DenseMatrix::Kron         │
                                └──────────────────────────────┘
```
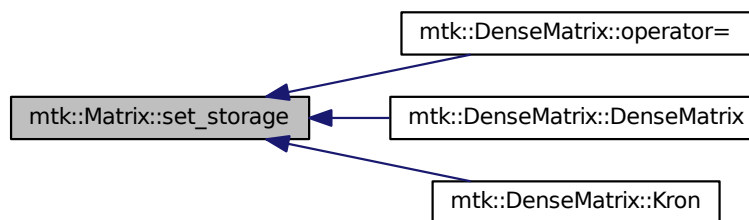
16.8.3.25 mtk::MatrixStorage mtk::Matrix::storage (   ) const

Definition at line 112 of file mtk_matrix.cc.

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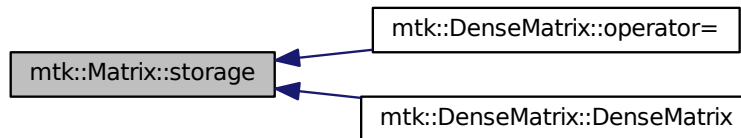
16.8.4 Member Data Documentation

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

Definition at line 288 of file mtk_matrix.h.

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

Definition at line 290 of file mtk_matrix.h.

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

Definition at line 286 of file mtk_matrix.h.

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

Definition at line 284 of file mtk_matrix.h.

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

Definition at line 285 of file mtk_matrix.h.

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

Definition at line 277 of file mtk_matrix.h.

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

Definition at line 275 of file mtk_matrix.h.

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

Definition at line 282 of file mtk_matrix.h.

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

Definition at line 280 of file mtk_matrix.h.

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

Definition at line 281 of file mtk_matrix.h.

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

Definition at line 274 of file mtk_matrix.h.

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

Definition at line 276 of file mtk_matrix.h.

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

Definition at line 279 of file mtk_matrix.h.

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

Definition at line 272 of file mtk_matrix.h.

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

Definition at line 289 of file mtk_matrix.h.

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

Definition at line 291 of file mtk_matrix.h.

16.8.4.17 MatrixStorage mtk::Matrix::storage_ `[private]`

Definition at line 270 of file mtk_matrix.h.

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

- include/mtk_matrix.h
- src/mtk_matrix.cc

16.9   mtk::Quad1D Class Reference

Implements a 1D mimetic quadrature.

```
#include <mtk_quad_1d.h>
```

Collaboration diagram for mtk::Quad1D:

```
                          ┌─────────────┐
                          │      T      │
                          ├─────────────┤
                          │             │
                          ├─────────────┤
                          │             │
                          └─────────────┘
                                 │
                                 │ +elements
                                 ◇
                      ┌──────────────────┐
                      │ std::vector< T > │
                      ├──────────────────┤
                      │                  │
                      ├──────────────────┤
                      │                  │
                      └──────────────────┘
                                ▲
                                │ < Real >
                      ┌────────────────────┐
                      │ std::vector< Real >│
                      ├────────────────────┤
                      │ + elements         │
                      ├────────────────────┤
                      │                    │
                      └────────────────────┘
                                │
                                │ -weights_
                                ◇
              ┌──────────────────────────────┐
              │       mtk::Quad1D            │
              ├──────────────────────────────┤
              │ - degree_approximation_     │
              ├──────────────────────────────┤
              │ + Quad1D()                  │
              │ + Quad1D()                  │
              │ + ~Quad1D()                 │
              │ + degree_approximation()    │
              │ + weights()                 │
              │ + Integrate()               │
              └──────────────────────────────┘
```
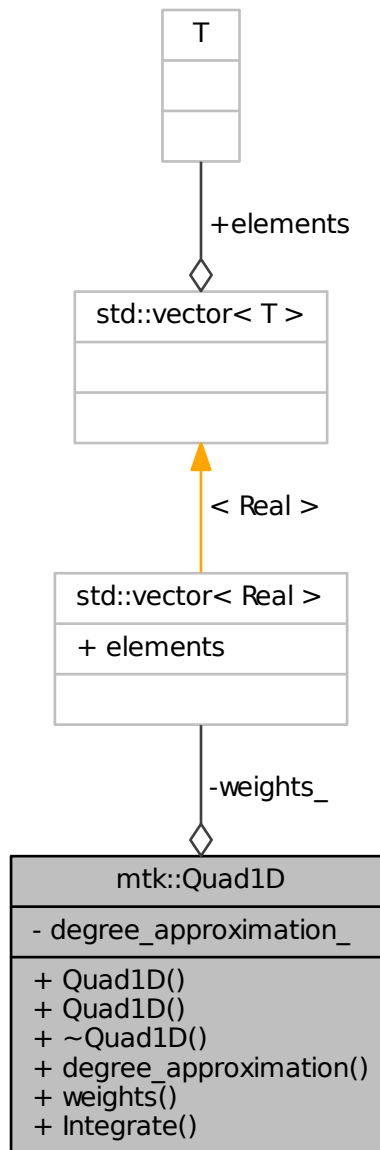
Public Member Functions

- Quad1D ()

  Default constructor.

- Quad1D (const Quad1D &quad)

  Copy constructor.

- ∼Quad1D ()

    Destructor.
- int degree_approximation () const

    Get the degree of interpolating polynomial per sub-interval of domain.
- Real ∗ weights () const

    Return collection of weights.
- Real Integrate (Real(∗Integrand)(Real xx), UniStgGrid1D grid)

    Mimetic integration routine.

Private Attributes

- int degree_approximation_

    Degree of the interpolating polynomial.
- std::vector< Real > weights_

    Collection of weights.

Friends

- std::ostream & operator<< (std::ostream &stream, Quad1D &in)

    Output stream operator for printing.

16.9.1 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.

16.9.2 Constructor & Destructor Documentation

16.9.2.1 mtk::Quad1D::Quad1D ( )

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

Parameters

in	*div*	Given quadrature.

16.9.2.3 mtk::Quad1D::∼Quad1D ( )

16.9.3 Member Function Documentation

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

Returns

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

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

Parameters

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

Returns

    Result of the integration.

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

Returns

    Collection of weights.

16.9.4 Friends And Related Function Documentation

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

16.9.5 Member Data Documentation

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

Definition at line 124 of file mtk_quad_1d.h.

16.9.5.2 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

16.10 mtk::Tools Class Reference

Tool manager class.

`#include <mtk_tools.h>`

Collaboration diagram for mtk::Tools:

```
┌─────────────────────────┐
│       mtk::Tools        │
├─────────────────────────┤
│ - test_number_          │
│ - begin_time_           │
├─────────────────────────┤
│ + Prevent()             │
│ + BeginTestNo()         │
│ + EndTestNo()           │
└─────────────────────────┘
```
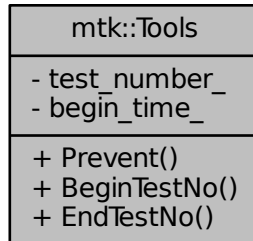
Static Public Member Functions

- static void Prevent (const bool condition, const char ∗fname, int lineno, const char ∗fxname)

    Enforces pre-conditions by preventing their complements from occur.
- static void BeginTestNo (const int &nn)

    Begins the execution of a test.
- static void EndTestNo (const int &nn)

    Ends the execution of a test.

Static Private Attributes

- static int test_number_

    Current test being executed.
- static clock_t begin_time_

    Elapsed time on current test.

16.10.1 Detailed Description

Basic tools to ensure execution correctness.

Definition at line 72 of file mtk_tools.h.

16.10.2 Member Function Documentation

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

Parameters

in	*nn*	Number of the test.

Definition at line 89 of file mtk_tools.cc.

Here is the call graph for this function:

16.10.2.2 void mtk::Tools::EndTestNo (const int & *nn*) `[static]`

Parameters

in	*nn*	Number of the test.

Definition at line 101 of file mtk_tools.cc.

Here is the call graph for this function:

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

See Also

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

Parameters

in	*condition*	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.

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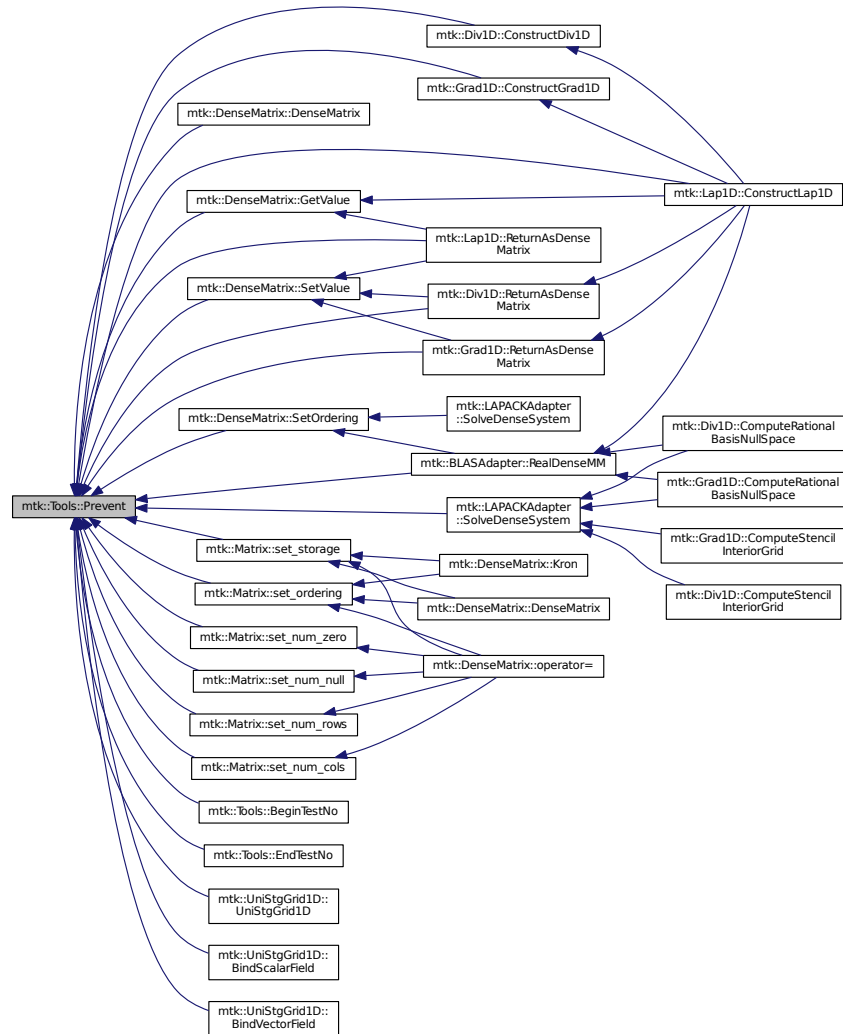
16.10.3 Member Data Documentation

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

Definition at line 106 of file mtk_tools.h.

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

Todo Check usage of static methods and private members.

Definition at line 104 of file mtk_tools.h.

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

- include/mtk_tools.h

- src/mtk_tools.cc

16.11 mtk::UniStgGrid1D Class Reference

Uniform 1D Staggered Grid.

```
#include <mtk_uni_stg_grid_1d.h>
```

Collaboration diagram for mtk::UniStgGrid1D:
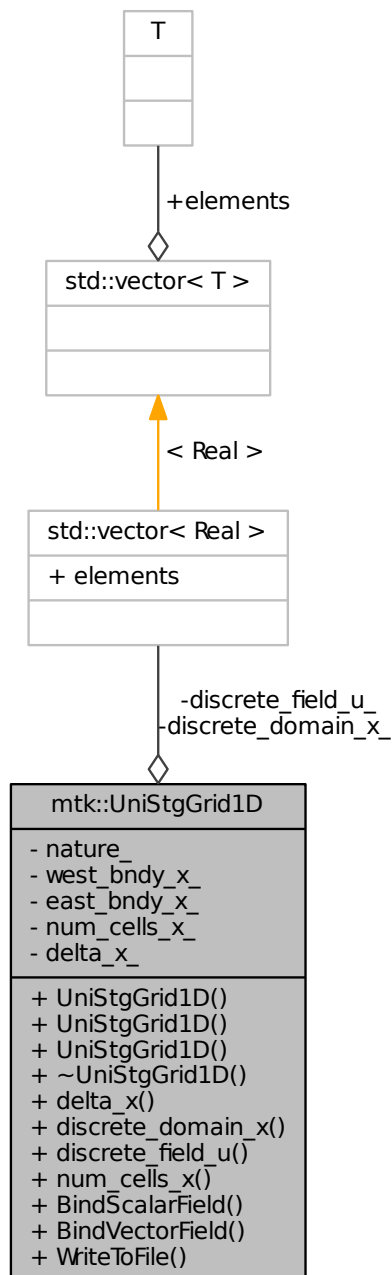
Public Member Functions

- 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 delta_x () const

    Provides access to the computed $ x $.

- Real ∗ discrete_domain_x ()

    Provides access to the grid spatial data.

- Real ∗ discrete_field_u ()

    Provides access to the grid field data.

- int num_cells_x () const

    Provides access to the number of cells of the grid.

- void BindScalarField (Real(∗ScalarField)(Real xx))

    Binds a given scalar field to the grid.

- void BindVectorField (Real(∗VectorField)(Real xx))

    Binds a given vector field to the grid.

- bool WriteToFile (std::string filename, std::string space_name, std::string field_name)

    Writes grid to a file compatible with Gnuplot 4.6.

Private Attributes

- FieldNature nature_

    Nature of the discrete field.

- std::vector< Real > discrete_domain_x_

    Array of spatial data.

- std::vector< Real > discrete_field_u_

    Array of field's data.

- Real west_bndy_x_

    West boundary spatial coordinate.

- Real east_bndy_x_

    East boundary spatial coordinate.

- Real num_cells_x_

    Number of cells discretizing the domain.

- Real delta_x_

    Produced Δx.

Friends

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

    Prints the grid as a tuple of arrays.

16.11.1 Detailed Description

Uniform 1D Staggered Grid.

Definition at line 77 of file mtk_uni_stg_grid_1d.h.

16.11.2 Constructor & Destructor Documentation

16.11.2.1 mtk::UniStgGrid1D::UniStgGrid1D ( )

Definition at line 97 of file mtk_uni_stg_grid_1d.cc.

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

Parameters

in	*grid*	Given grid.

Definition at line 106 of file mtk_uni_stg_grid_1d.cc.

16.11.2.3 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 122 of file mtk_uni_stg_grid_1d.cc.

Here is the call graph for this function:
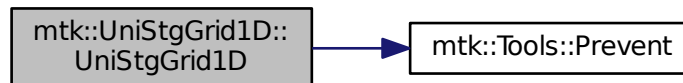
16.11.2.4 mtk::UniStgGrid1D::∼UniStgGrid1D ( )

Definition at line 142 of file mtk_uni_stg_grid_1d.cc.

16.11.3 Member Function Documentation

16.11.3.1 void mtk::UniStgGrid1D::BindScalarField (Real(∗)(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 164 of file mtk_uni_stg_grid_1d.cc.

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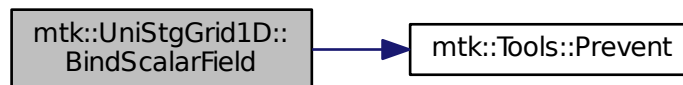
16.11.3.2 void mtk::UniStgGrid1D::BindVectorField (Real(∗)(Real xx) *VectorField*)

We assume the field to be of the form:

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

Parameters

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

1. Create collection of spatial coordinates.

2. Create collection of field samples.

Definition at line 200 of file mtk_uni_stg_grid_1d.cc.

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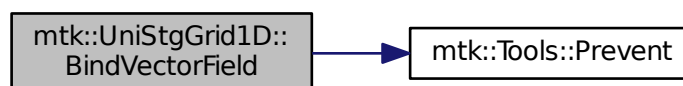
16.11.3.3   mtk::Real mtk::UniStgGrid1D::delta_x (   ) const

Returns

Computed $ x $.

Definition at line 144 of file mtk_uni_stg_grid_1d.cc.

Here is the caller graph for this function:
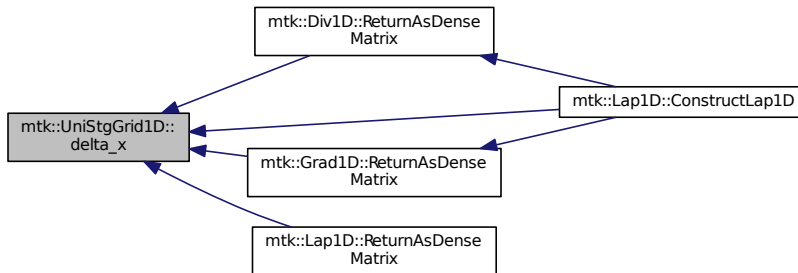
16.11.3.4   mtk::Real ∗ mtk::UniStgGrid1D::discrete_domain_x (   )

Returns

Pointer to the spatial data.

Definition at line 149 of file mtk_uni_stg_grid_1d.cc.

16.11.3.5   mtk::Real ∗ mtk::UniStgGrid1D::discrete_field_u (   )

Returns

Pointer to the field data.

Definition at line 154 of file mtk_uni_stg_grid_1d.cc.

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

Returns

Number of cells of the grid.

Definition at line 159 of file mtk_uni_stg_grid_1d.cc.

Here is the caller graph for this function:
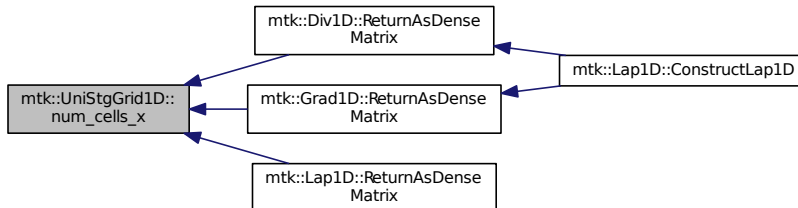
16.11.3.7 **bool mtk::UniStgGrid1D::WriteToFile (std::string *filename,* std::string *space_name,* std::string *field_name*)**

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 228 of file mtk_uni_stg_grid_1d.cc.

16.11.4 **Friends And Related Function Documentation**

16.11.4.1 **std::ostream& operator<< (std::ostream & *stream,* mtk::UniStgGrid1D & *in*)** `[friend]`

1. Print spatial coordinates.

Definition at line 68 of file mtk_uni_stg_grid_1d.cc.

16.11.5 **Member Data Documentation**

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

Definition at line 182 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 176 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 177 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 180 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 174 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 181 of file mtk_uni_stg_grid_1d.h.

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

Definition at line 179 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

Chapter 17

File Documentation

17.1 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:

```
┌─────────────────────┐
│ examples/poisson_1d │
│    /poisson_1d.cc   │
└─────────────────────┘
           │
           ▼
      ┌──────────┐
      │ iostream │
      └──────────┘
```
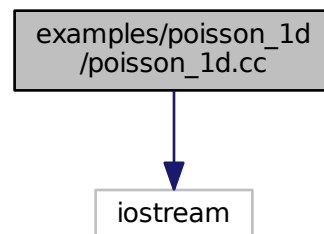
Functions

- int main ()

17.1.1 Detailed Description

We solve:

$$-\nabla^2 u(x) = s(x),$$

for $x \in \Omega = [0, 1]$.

The source term function is defined as

$$s(x) = \frac{-\lambda^2 \exp(\lambda x)}{\exp(\lambda) - 1}$$

where $\lambda = -1$ is a parameter.

We consider Robin's boundary conditions of the form:

$$\alpha u(a) - \beta u'(a) = \text{west\_bndy\_cond\_val},$$

$$\alpha u(b) + \beta u'(b) = \text{east\_bndy\_cond\_val}.$$

The analytical solution for this problem is given by

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

Author

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

Definition in file poisson_1d.cc.

17.1.2 Function Documentation

17.1.2.1 int main ( )

Definition at line 190 of file poisson_1d.cc.

17.2 poisson_1d.cc

```
00001
00042 /*
00043 Copyright (C) 2015, Computational Science Research Center, San Diego State
00044 University. All rights reserved.
00045
00046 Redistribution and use in source and binary forms, with or without modification,
00047 are permitted provided that the following conditions are met:
00048
00049 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00050 and a copy of the modified files should be reported once modifications are
00051 completed. Documentation related to said modifications should be included.
00052
00053 2. Redistributions of source code must be done through direct
00054 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00055
00056 3. Redistributions of source code must retain the above copyright notice, this
00057 list of conditions and the following disclaimer.
00058
00059 4. Redistributions in binary form must reproduce the above copyright notice,
00060 this list of conditions and the following disclaimer in the documentation and/or
00061 other materials provided with the distribution.
00062
00063 5. Usage of the binary form on proprietary applications shall require explicit
00064 prior written permission from the the copyright holders.
00065
```

```
00066 6. Neither the name of the copyright holder nor the names of its contributors
00067 may be used to endorse or promote products derived from this software without
00068 specific prior written permission.
00069
00070 The copyright holders provide no reassurances that the source code provided does
00071 not infringe any patent, copyright, or any other intellectual property rights of
00072 third parties. The copyright holders disclaim any liability to any recipient for
00073 claims brought against recipient by any third party for infringement of that
00074 parties intellectual property rights.
00075
00076 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00077 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00078 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00079 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00080 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00081 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00082 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00083 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00084 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00085 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00086 */
00087
00088 #if __cplusplus == 201103L
00089
00090 #include <iostream>
00091 #include <fstream>
00092 #include <cmath>
00093
00094 #include "mtk.h"
00095
00096 mtk::Real Source(mtk::Real xx) {
00097
00098   mtk::Real lambda = -1.0;
00099
00100   return -lambda*lambda*exp(lambda*xx)/(exp(lambda) - 1.0);
00101 }
00102
00103 mtk::Real KnownSolution(mtk::Real xx) {
00104
00105   mtk::Real lambda = -1.0;
00106
00107   return (exp(lambda*xx) - 1.0)/(exp(lambda) - 1.0);
00108 }
00109
00110 int main () {
00111
00112   std::cout << "Example: Poisson Equation on a 1D Uniform Staggered Grid ";
00113   std::cout << "with Robin BCs." << std::endl;
00114
00116
00117   mtk::Real lambda = -1.0;
00118   mtk::Real alpha = -exp(lambda);
00119   mtk::Real beta = (exp(lambda) - 1.0)/lambda;
00120   mtk::Real west_bndy_value = -1.0;
00121   mtk::Real east_bndy_value = 0.0;
00122
00124
00125   mtk::Real west_bndy_x = 0.0;
00126   mtk::Real east_bndy_x = 1.0;
00127   int num_cells_x = 5;
00128
00129   mtk::UniStgGrid1D comp_sol(west_bndy_x, east_bndy_x, num_cells_x);
00130
00132
00133   int order_of_accuracy = 2;
00134
00135   mtk::Grad1D grad; // Mimetic gradient operator.
00136
00137   mtk::Lap1D lap; // Mimetic gradient operator.
00138
00139   mtk::Real mimetic_threshold = 1e-6;
00140
00141   if (!lap.ConstructLap1D(order_of_accuracy, mimetic_threshold)) {
00142     std::cerr << "Mimetic lap could not be built." << std::endl;
00143     return EXIT_FAILURE;
00144   }
00145
00146   mtk::DenseMatrix lapm(lap.ReturnAsDenseMatrix(comp_sol));
00147
00148   std::cout << "Mimetic Laplacian operator: " << std::endl;
00149   std::cout << lapm << std::endl;
```

```
00150
00151    if (!grad.ConstructGrad1D(order_of_accuracy, mimetic_threshold)) {
00152      std::cerr << "Mimetic grad could not be built." << std::endl;
00153      return EXIT_FAILURE;
00154    }
00155
00156    mtk::DenseMatrix gradm(grad.ReturnAsDenseMatrix(comp_sol));
00157
00158    std::cout << "Mimetic gradient operator: " << std::endl;
00159    std::cout << gradm << std::endl;
00160
00162    mtk::UniStgGrid1D source(west_bndy_x, east_bndy_x, num_cells_x);
00164
00165    source.BindScalarField(Source);
00166
00167    std::cout << source << std::endl;
00168
00169    source.WriteToFile("poisson_1d_source.dat", "x", "s(x)");
00170
00172
00174
00176
00177    mtk::UniStgGrid1D known_sol(west_bndy_x, east_bndy_x, num_cells_x);
00178
00179    known_sol.BindScalarField(KnownSolution);
00180
00181    std::cout << known_sol << std::endl;
00182
00183    known_sol.WriteToFile("poisson_1d_known_sol.dat", "x", "u(x)");
00184  }
00185
00186  #else
00187  #include <iostream>
00188  using std::cout;
00189  using std::endl;
00190  int main () {
00191    cout << "This code HAS to be compiled with support for C++11." << endl;
00192    cout << "Exiting..." << endl;
00193    return EXIT_SUCCESS;
00194  }
00195  #endif
```

17.3 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_grad_1d.h"
#include "mtk_div_1d.h"
#include "mtk_lap_1d.h"
#include "mtk_quad_1d.h"
```

Include dependency graph for mtk.h:
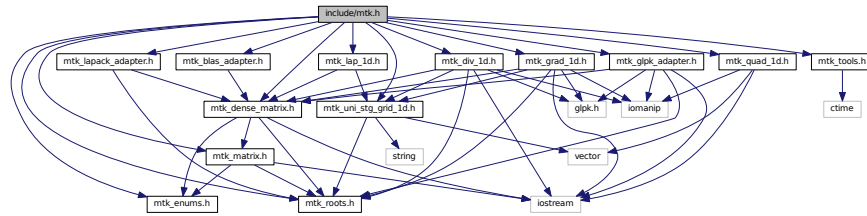
17.3.1 Detailed Description

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.

17.4 mtk.h

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed. Documentation related to said modifications should be included.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
```

```
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00360 #ifndef MTK_INCLUDE_MTK_H_
00361 #define MTK_INCLUDE_MTK_H_
00362
00370 #include "mtk_roots.h"
00371
00379 #include "mtk_enums.h"
00380
00388 #include "mtk_tools.h"
00389
00397 #include "mtk_matrix.h"
00398 #include "mtk_dense_matrix.h"
00399
00407 #include "mtk_blas_adapter.h"
00408 #include "mtk_lapack_adapter.h"
00409 #include "mtk_glpk_adapter.h"
00410
00418 #include "mtk_uni_stg_grid_1d.h"
00419
00427 #include "mtk_grad_1d.h"
00428 #include "mtk_div_1d.h"
00429 #include "mtk_lap_1d.h"
00430 #include "mtk_quad_1d.h"
00431
00432 #endif // End of: MTK_INCLUDE_MTK_H_
```

17.5   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:
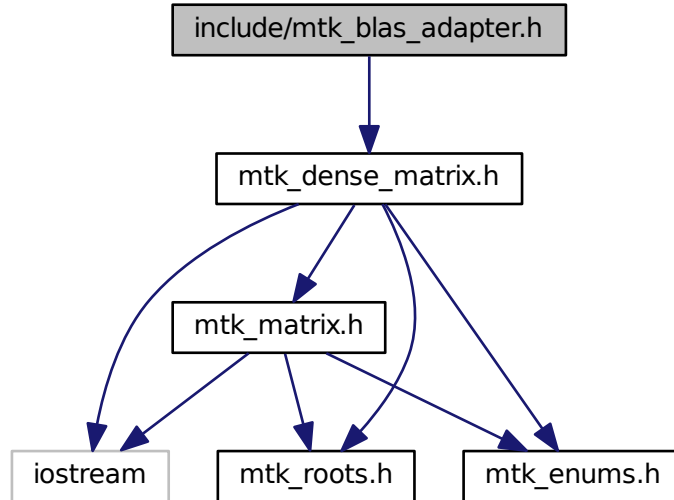
This graph shows which files directly or indirectly include this file:
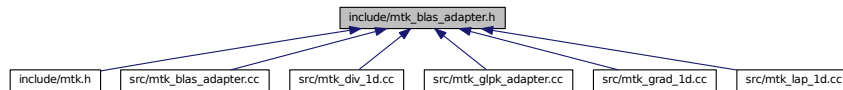
Classes

- class mtk::BLASAdapter

  Adapter class for the BLAS API.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.5.1 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.

17.6 mtk_blas_adapter.h

```
00001
00024 /*
00025 Copyright (C) 2015, Computational Science Research Center, San Diego State
00026 University. All rights reserved.
00027
00028 Redistribution and use in source and binary forms, with or without modification,
00029 are permitted provided that the following conditions are met:
00030
00031 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00032 and a copy of the modified files should be reported once modifications are
00033 completed. Documentation related to said modifications should be included.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions of source code must retain the above copyright notice, this
00039 list of conditions and the following disclaimer.
00040
00041 4. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 5. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders.
00047
00048 6. Neither the name of the copyright holder nor the names of its contributors
00049 may be used to endorse or promote products derived from this software without
00050 specific prior written permission.
00051
00052 The copyright holders provide no reassurances that the source code provided does
00053 not infringe any patent, copyright, or any other intellectual property rights of
00054 third parties. The copyright holders disclaim any liability to any recipient for
00055 claims brought against recipient by any third party for infringement of that
00056 parties intellectual property rights.
00057
00058 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00059 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00060 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00061 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00062 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00063 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00064 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00065 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00066 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #ifndef MTK_INCLUDE_BLAS_ADAPTER_H_
00071 #define MTK_INCLUDE_BLAS_ADAPTER_H_
00072
00073 #include "mtk_dense_matrix.h"
```

```
00074
00075 namespace mtk {
00076
00096 class BLASAdapter {
00097  public:
00104   static Real RealNRM2(Real *in, int &in_length);
00105
00123   static void RealDenseMV(Real &alpha,
00124                           DenseMatrix &aa,
00125                           Real *xx,
00126                           Real &beta,
00127                           Real *yy);
00128
00143   static DenseMatrix RealDenseMM(DenseMatrix &aa,
00144 };    DenseMatrix &bb);
00145 }
00146 #endif  // End of: MTK_INCLUDE_BLAS_ADAPTER_H_
```

17.7 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:
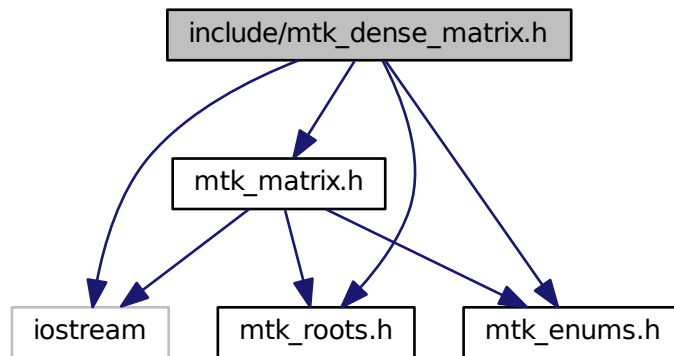
This graph shows which files directly or indirectly include this file:
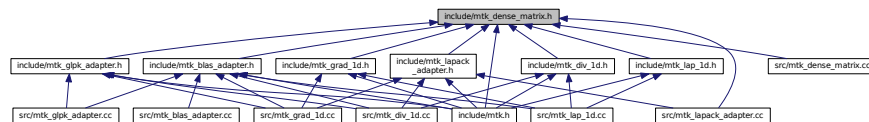
Classes

- class [mtk::DenseMatrix](#)

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

Namespaces

- [mtk](#)

    Mimetic Methods Toolkit namespace.

17.7.1 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

Todo Add sparse matrices support: BANDED and CRS.

Todo Contemplate manipulation of sparse metrics.

Todo Implement Kronecker product using the BLAS.

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](#).

17.8 mtk_dense_matrix.h

```
00001
00029 /*
00030 Copyright (C) 2015, Computational Science Research Center, San Diego State
00031 University. All rights reserved.
00032
00033 Redistribution and use in source and binary forms, with or without modification,
00034 are permitted provided that the following conditions are met:
00035
00036 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00037 and a copy of the modified files should be reported once modifications are
00038 completed. Documentation related to said modifications should be included.
00039
00040 2. Redistributions of source code must be done through direct
00041 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00042
00043 3. Redistributions of source code must retain the above copyright notice, this
```

```
00044 list of conditions and the following disclaimer.
00045
00046 4. Redistributions in binary form must reproduce the above copyright notice,
00047 this list of conditions and the following disclaimer in the documentation and/or
00048 other materials provided with the distribution.
00049
00050 5. Usage of the binary form on proprietary applications shall require explicit
00051 prior written permission from the the copyright holders.
00052
00053 6. Neither the name of the copyright holder nor the names of its contributors
00054 may be used to endorse or promote products derived from this software without
00055 specific prior written permission.
00056
00057 The copyright holders provide no reassurances that the source code provided does
00058 not infringe any patent, copyright, or any other intellectual property rights of
00059 third parties. The copyright holders disclaim any liability to any recipient for
00060 claims brought against recipient by any third party for infringement of that
00061 parties intellectual property rights.
00062
00063 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00064 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00065 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00066 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00067 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00068 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00069 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00070 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00071 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00072 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00073 */
00074
00075 #ifndef MTK_INCLUDE_DENSE_MATRIX_H_
00076 #define MTK_INCLUDE_DENSE_MATRIX_H_
00077
00078 #include <iostream>
00079
00080 #include "mtk_roots.h"
00081 #include "mtk_enums.h"
00082 #include "mtk_matrix.h"
00083
00084 namespace mtk {
00085
00098 class DenseMatrix {
00099  public:
00101   friend std::ostream& operator <<(std::ostream &stream, DenseMatrix &in);
00102
00104   DenseMatrix& operator =(const DenseMatrix &in);
00105
00107   DenseMatrix();
00108
00114   DenseMatrix(const DenseMatrix &in);
00115
00124   DenseMatrix(const int &num_rows, const int &num_cols);
00125
00151   DenseMatrix(const int &rank, const bool &padded, const bool &transpose);
00152
00186   DenseMatrix(const Real *gen,
00187               const int &gen_length,
00188               const int &pro_length,
00189               const bool &transpose);
00190
00192   ~DenseMatrix();
00193
00199   Matrix matrix_properties() const;
00200
00206   int num_rows() const;
00207
00213   int num_cols() const;
00214
00220   Real* data() const;
00221
00229   void SetOrdering(mtk::MatrixOrdering oo);
00230
00239   Real GetValue(const int &row_coord, const int &col_coord) const;
00240
00248   void SetValue(const int &row_coord,
00249                 const int &col_coord,
00250                 const Real &val);
00251
00253   void Transpose();
00254
```

```
00256   void OrderRowMajor();
00257
00259   void OrderColMajor();
00260
00269   static DenseMatrix Kron(const DenseMatrix &aa, const
     DenseMatrix &bb);
00270
00271 private:
00272   Matrix matrix_properties_;
00273
00274   Real *data_;
00275 };
00276 }
00277 #endif  // End of: MTK_INCLUDE_MTK_DENSE_MATRIX_H_
```

17.9   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:
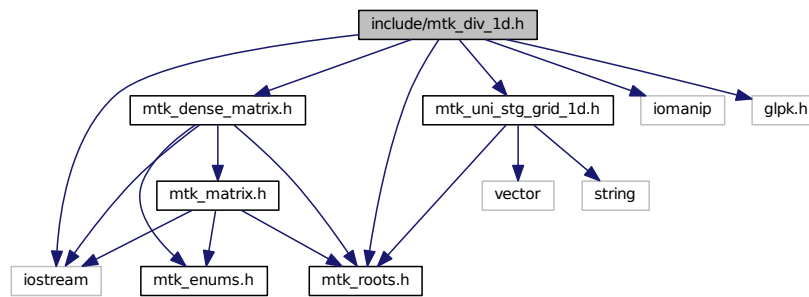
This graph shows which files directly or indirectly include this file:
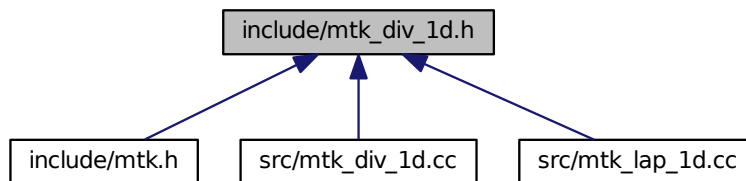
Classes

- class mtk::Div1D

  Implements a 1D mimetic divergence operator.

Namespaces

- mtk

  Mimetic Methods Toolkit namespace.

17.9.1 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.

17.10 mtk_div_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```

```
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_DIV_1D_H_
00058 #define MTK_INCLUDE_DIV_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Div1D {
00082  public:
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    Real* weights_crs(void);
00113
00119    Real* weights_cbs(void);
00120
00126    DenseMatrix ReturnAsDenseMatrix(const
00127  UniStgGrid1D &grid);
00127
00128  private:
00134    bool ComputeStencilInteriorGrid(void);
00135
00142    bool ComputeRationalBasisNullSpace(void);
00143
00149    bool ComputePreliminaryApproximations(void);
00150
00156    bool ComputeWeights(void);
00157
00163    bool ComputeStencilBoundaryGrid(void);
00164
00170    bool AssembleOperator(void);
00171
00172    int order_accuracy_;
00173    int dim_null_;
00174    int num_bndy_coeffs_;
00175    int divergence_length_;
00176
00177    int minrow_;
00178    int row_;
00179
00180    mtk::DenseMatrix rat_basis_null_space_;
00181
00182    Real *coeffs_interior_;
00183    Real *prem_apps_;
00184    Real *weights_crs_;
00185    Real *weights_cbs_;
00186    Real *mim_bndy_;
00187    Real *divergence_;
00188
00189    Real mimetic_threshold_;
00190 };
00191 }
00192 #endif  // End of: MTK_INCLUDE_DIV_1D_H_
```

17.11 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
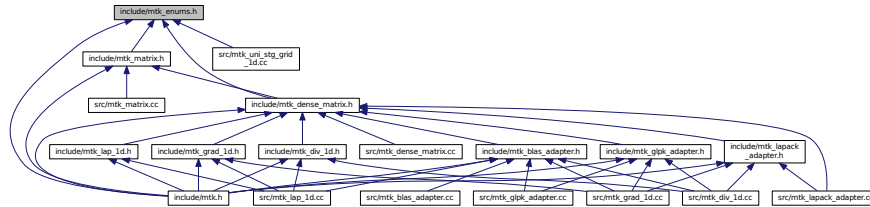
  Mimetic Methods Toolkit namespace.

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.

17.11.1 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.

17.12 mtk_enums.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
```

```
00020 and a copy of the modified files should be reported once modifications are
00021 completed. Documentation related to said modifications should be included.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
00028
00029 4. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_ENUMS_H_
00059 #define MTK_INCLUDE_ENUMS_H_
00060
00061 namespace mtk {
00062
00077 enum MatrixStorage {
00078   DENSE,
00079   BANDED,
00080   CRS
00081 };
00082
00095 enum MatrixOrdering {
00096   ROW_MAJOR,
00097   COL_MAJOR
00098 };
00099
00113 enum FieldNature {
00114   SCALAR,
00115   VECTOR
00116 };
00117 }
00118 #endif  // End of: MTK_INCLUDE_ENUMS_H_
```

17.13   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:
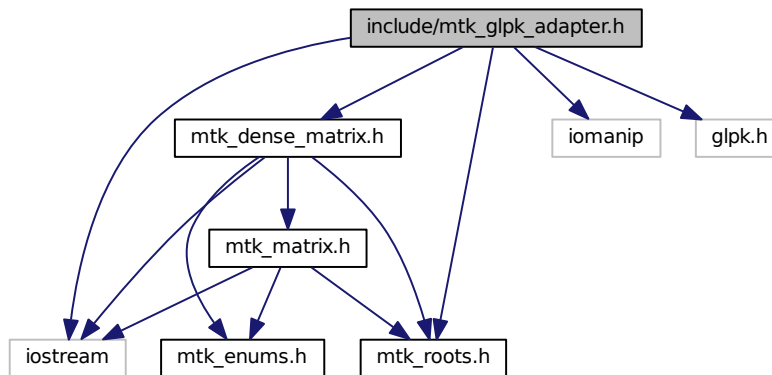
This graph shows which files directly or indirectly include this file:
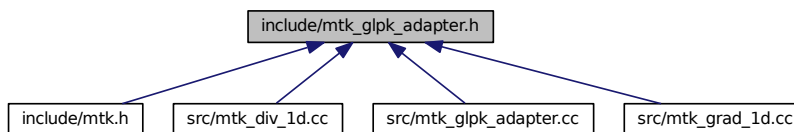
Classes

- class mtk::GLPKAdapter

    Adapter class for the GLPK API.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.13.1 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 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.

17.14 mtk_glpk_adapter.h

```
00001
00019 /*
00020 Copyright (C) 2015, Computational Science Research Center, San Diego State
00021 University. All rights reserved.
00022
00023 Redistribution and use in source and binary forms, with or without modification,
00024 are permitted provided that the following conditions are met:
00025
00026 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00027 and a copy of the modified files should be reported once modifications are
00028 completed. Documentation related to said modifications should be included.
00029
00030 2. Redistributions of source code must be done through direct
00031 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00032
00033 3. Redistributions of source code must retain the above copyright notice, this
00034 list of conditions and the following disclaimer.
00035
00036 4. Redistributions in binary form must reproduce the above copyright notice,
00037 this list of conditions and the following disclaimer in the documentation and/or
00038 other materials provided with the distribution.
00039
00040 5. Usage of the binary form on proprietary applications shall require explicit
00041 prior written permission from the the copyright holders.
00042
00043 6. Neither the name of the copyright holder nor the names of its contributors
00044 may be used to endorse or promote products derived from this software without
00045 specific prior written permission.
00046
00047 The copyright holders provide no reassurances that the source code provided does
00048 not infringe any patent, copyright, or any other intellectual property rights of
00049 third parties. The copyright holders disclaim any liability to any recipient for
00050 claims brought against recipient by any third party for infringement of that
00051 parties intellectual property rights.
00052
00053 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00054 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00055 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00056 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00057 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00058 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00059 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00060 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00061 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00062 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00063 */
00064
00065 #ifndef MTK_INCLUDE_GLPK_ADAPTER_H_
00066 #define MTK_INCLUDE_GLPK_ADAPTER_H_
00067
00068 #include <iostream>
00069 #include <iomanip>
00070
00071 #include "glpk.h"
00072
00073 #include "mtk_roots.h"
00074 #include "mtk_dense_matrix.h"
00075
00076 namespace mtk {
00077
00101 class GLPKAdapter {
```

```
00102  public:
00121   static mtk::Real SolveSimplexAndCompare(
    mtk::Real *A,
00122                                            int nrows,
00123                                            int ncols,
00124                                            int kk,
00125                                            mtk::Real *hh,
00126                                            mtk::Real *qq,
00127                                            int robjective,
00128                                            mtk::Real mimetic_tol,
00129                                            int copy);
00130 };
00131 }
00132 #endif  // End of: MTK_INCLUDE_MTK_GLPK_ADAPTER_H_
```

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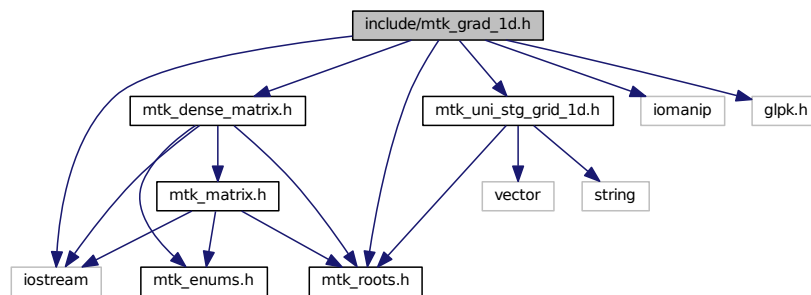
This graph shows which files directly or indirectly include this file:
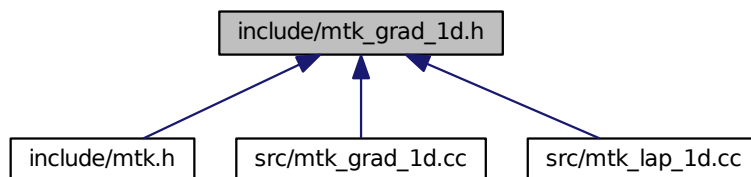
Classes

- class mtk::Grad1D

    Implements a 1D mimetic gradient operator.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.15.1 Detailed Description

This class implements a 1D gradient 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_grad_1d.h.

17.16 mtk_grad_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```

```
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_GRAD_1D_H_
00058 #define MTK_INCLUDE_GRAD_1D_H_
00059
00060 #include <iostream>
00061 #include <iomanip>
00062
00063 #include "glpk.h"
00064
00065 #include "mtk_roots.h"
00066 #include "mtk_dense_matrix.h"
00067 #include "mtk_uni_stg_grid_1d.h"
00068
00069 namespace mtk {
00070
00081 class Grad1D {
00082  public:
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    Real* weights_crs(void);
00113
00119    Real* weights_cbs(void);
00120
00126    DenseMatrix ReturnAsDenseMatrix(const
00127    UniStgGrid1D &grid);
00127
00128  private:
00134    bool ComputeStencilInteriorGrid(void);
00135
00142    bool ComputeRationalBasisNullSpace(void);
00143
00149    bool ComputePreliminaryApproximations(void);
00150
00156    bool ComputeWeights(void);
00157
00163    bool ComputeStencilBoundaryGrid(void);
00164
00170    bool AssembleOperator(void);
00171
00172    int order_accuracy_;
00173    int dim_null_;
00174    int num_bndy_approxs_;
00175    int num_bndy_coeffs_;
00176    int gradient_length_;
00177
00178    int minrow_;
00179    int row_;
00180
00181    mtk::DenseMatrix rat_basis_null_space_;
00182
00183    Real *coeffs_interior_;
00184    Real *prem_apps_;
00185    Real *weights_crs_;
00186    Real *weights_cbs_;
00187    Real *mim_bndy_;
00188    Real *gradient_;
00189
00190    Real mimetic_threshold_;
00191 };
00192 }
00193 #endif  // End of: MTK_INCLUDE_GRAD_1D_H_
```

17.17 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:
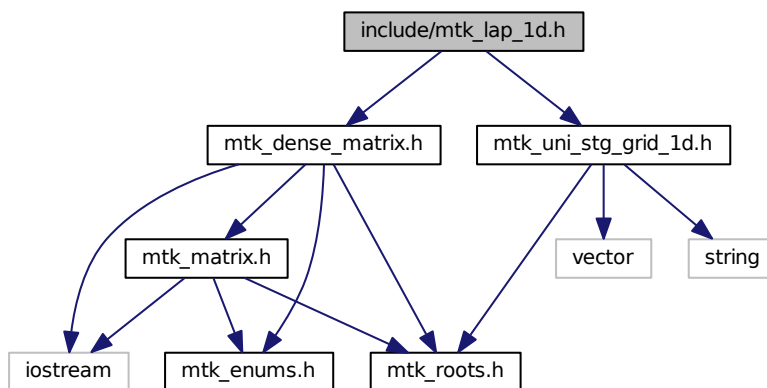
This graph shows which files directly or indirectly include this file:
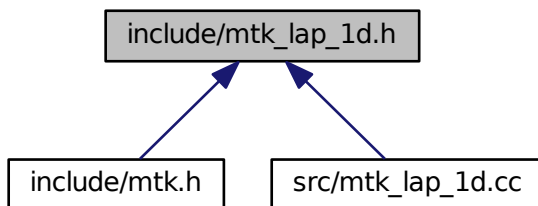
Classes

- class mtk::Lap1D

  Implements a 1D mimetic Laplacian operator.

Namespaces

- mtk

Mimetic Methods Toolkit namespace.

17.17.1 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.

17.18 mtk_lap_1d.h

```
00001
00011 /*
00012 Copyright (C) 2015, Computational Science Research Center, San Diego State
00013 University. All rights reserved.
00014
00015 Redistribution and use in source and binary forms, with or without modification,
00016 are permitted provided that the following conditions are met:
00017
00018 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00019 and a copy of the modified files should be reported once modifications are
00020 completed. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
00032 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #ifndef MTK_INCLUDE_LAP_1D_H_
00058 #define MTK_INCLUDE_LAP_1D_H_
00059
00060 #include "mtk_dense_matrix.h"
00061
00062 #include "mtk_uni_stg_grid_1d.h"
00063
00064 namespace mtk {
```

```
00065
00076 class Lap1D {
00077  public:
00079    friend std::ostream& operator <<(std::ostream& stream, Lap1D &in);
00080
00082    Lap1D();
00083
00089    Lap1D(const Lap1D &lap);
00090
00092    ~Lap1D();
00093
00099    bool ConstructLap1D(int order_accuracy = kDefaultOrderAccuracy,
00100                        Real mimetic_threshold = kDefaultMimeticThreshold);
00101
00107    DenseMatrix ReturnAsDenseMatrix(const
      UniStgGrid1D &grid);
00108
00114    mtk::Real* Data(const UniStgGrid1D &grid);
00115
00116  private:
00117    int order_accuracy_;
00118    int laplacian_length_;
00119
00120    Real *laplacian_;
00121
00122    Real mimetic_threshold_;
00123 };
00124 }
00125 #endif  // End of: MTK_INCLUDE_LAP_1D_H_
```

17.19   include/mtk_lapack_adapter.h File Reference

Adapter class for the LAPACK API.

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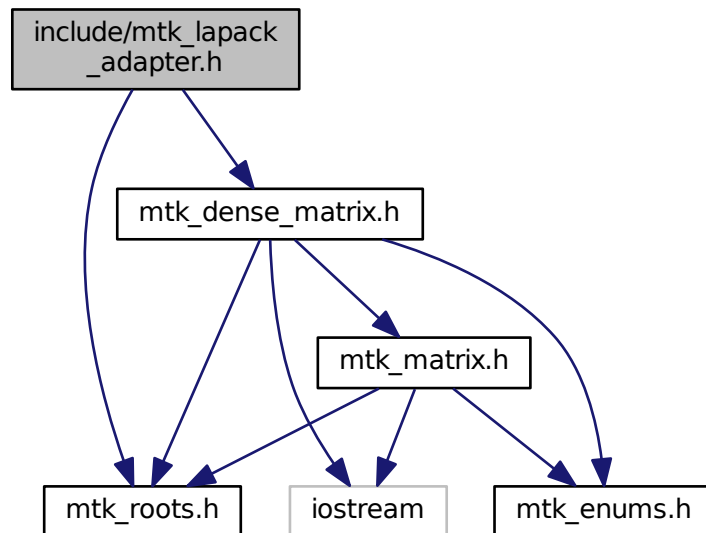
This graph shows which files directly or indirectly include this file:
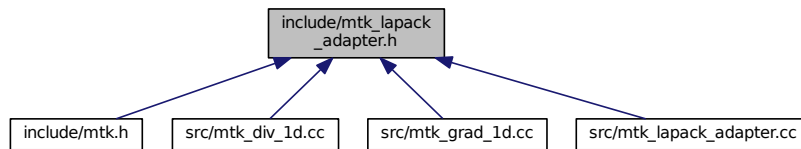
Classes

- class mtk::LAPACKAdapter

    Adapter class for the LAPACK API.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.19.1 Detailed Description

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

The **LAPACK** 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.

17.20 mtk_lapack_adapter.h

```
00001
00018 /*
00019 Copyright (C) 2015, Computational Science Research Center, San Diego State
00020 University. All rights reserved.
00021
00022 Redistribution and use in source and binary forms, with or without modification,
00023 are permitted provided that the following conditions are met:
00024
00025 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00026 and a copy of the modified files should be reported once modifications are
00027 completed. Documentation related to said modifications should be included.
```

```
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 of source code must retain the above copyright notice, this
00033 list of conditions and the following disclaimer.
00034
00035 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders.
00041
00042 6. Neither the name of the copyright holder nor the names of its contributors
00043 may be used to endorse or promote products derived from this software without
00044 specific prior written permission.
00045
00046 The copyright holders provide no reassurances that the source code provided does
00047 not infringe any patent, copyright, or any other intellectual property rights of
00048 third parties. The copyright holders disclaim any liability to any recipient for
00049 claims brought against recipient by any third party for infringement of that
00050 parties intellectual property rights.
00051
00052 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00053 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00054 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00055 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00056 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00057 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00058 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00059 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00060 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00061 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00062 */
00063
00064 #ifndef MTK_INCLUDE_LAPACK_ADAPTER_H_
00065 #define MTK_INCLUDE_LAPACK_ADAPTER_H_
00066
00067 #include "mtk_roots.h"
00068 #include "mtk_dense_matrix.h"
00069
00070 namespace mtk {
00071
00089 class LAPACKAdapter {
00090  public:
00101   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00102                               mtk::Real *rhs);
00103
00114   static int SolveDenseSystem(mtk::DenseMatrix &mm,
00115                               mtk::DenseMatrix &rr);
00116
00128   static int SolveRectangularDenseSystem(const
00     mtk::DenseMatrix &aa,
00129                                          mtk::Real *ob_,
00130                                          int ob_ld_);
00131
00143   static mtk::DenseMatrix QRFactorDenseMatrix(
00     DenseMatrix &matrix);
00144 };
00145 }
00146 #endif  // End of: MTK_INCLUDE_LAPACK_ADAPTER_H_
```

17.21   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.

17.21.1 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](#).

17.22 mtk_matrix.h

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_MATRIX_H_
00057 #define MTK_INCLUDE_MATRIX_H_
00058
00059 #include <iostream>
00060
00061 #include "mtk_roots.h"
00062 #include "mtk_enums.h"
00063
00064 namespace mtk {
00065
00075 class Matrix {
00076  public:
00078   Matrix();
00079
00085   Matrix(const Matrix &in);
00086
00088   ~Matrix();
00089
00091   MatrixStorage storage() const;
00092
00094   MatrixOrdering ordering() const;
00095
00101   int num_rows() const;
00102
00108   int num_cols() const;
00109
00115   int num_values() const;
00116
00126   int ld() const;
00127
00133   int num_zero() const;
```

```
00134
00140    int num_non_zero() const;
00141
00149    int num_null() const;
00150
00158    int num_non_null() const;
00159
00165    int kl() const;
00166
00172    int ku() const;
00173
00179    int bandwidth() const;
00180
00188    Real abs_density() const;
00189
00197    Real rel_density() const;
00198
00206    Real abs_sparsity() const;
00207
00215    Real rel_sparsity() const;
00216
00224    void set_storage(const MatrixStorage &tt);
00225
00233    void set_ordering(const MatrixOrdering &oo);
00234
00240    void set_num_rows(int num_rows);
00241
00247    void set_num_cols(int num_cols);
00248
00254    void set_num_zero(int in);
00255
00261    void set_num_null(int in);
00262
00264    void IncreaseNumZero();
00265
00267    void IncreaseNumNull();
00268
00269  private:
00270    MatrixStorage storage_;
00271
00272    MatrixOrdering ordering_;
00273
00274    int num_rows_;
00275    int num_cols_;
00276    int num_values_;
00277    int ld_;
00278
00279    int num_zero_;
00280    int num_non_zero_;
00281    int num_null_;
00282    int num_non_null_;
00283
00284    int kl_;
00285    int ku_;
00286    int bandwidth_;
00287
00288    Real abs_density_;
00289    Real rel_density_;
00290    Real abs_sparsity_;
00291    Real rel_sparsity_;
00292 };
00293 }
00294 #endif  // End of: MTK_INCLUDE_MATRIX_H_
```

17.23   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
```
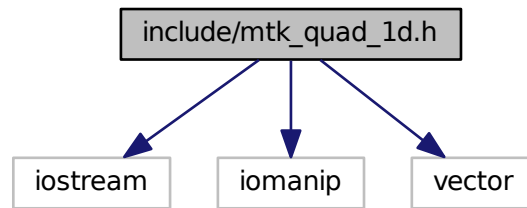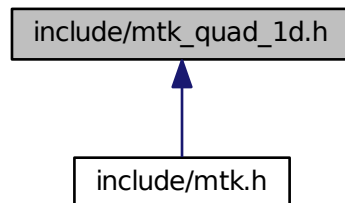
Classes

- class mtk::Quad1D

    Implements a 1D mimetic quadrature.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.23.1 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.

17.24 mtk_quad_1d.h

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed. Documentation related to said modifications should be included.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #ifndef MTK_INCLUDE_QUAD_1D_H_
00062 #define MTK_INCLUDE_QUAD_1D_H_
00063
00064 #include <iostream>
00065 #include <iomanip>
00066
00067 #include <vector>
00068
00069 namespace mtk {
00070
00081 class Quad1D {
00082  public:
00084    friend std::ostream& operator <<(std::ostream& stream, Quad1D &in);
00085
00087    Quad1D();
00088
00094    Quad1D(const Quad1D &quad);
00095
```

```
00097   ~Quad1D();
00098
00104   int degree_approximation() const;
00105
00111   Real *weights() const;
00112
00121   Real Integrate(Real (*Integrand)(Real xx), UniStgGrid1D grid);
00122
00123 private:
00124   int degree_approximation_;
00125
00126   std::vector<Real> weights_;
00127 };
00128 }
00129 #endif  // End of: MTK_INCLUDE_QUAD_1D_H_
```

17.25   include/mtk_roots.h File Reference

Fundamental definitions to be used across all classes of the MTK.

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

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Typedefs

- typedef float mtk::Real

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

Variables

- const float mtk::kZero {0.0f}

    MTK's zero defined according to selective compilation.

- const float mtk::kOne {1.0f}

    MTK's one defined according to selective compilation.

- const float mtk::kDefaultTolerance {1e-7f}

    Considered tolerance for comparisons in numerical methods.

- const int mtk::kDefaultOrderAccuracy {2}

    Default order of accuracy for mimetic operators.

- const float mtk::kDefaultMimeticThreshold {1.e-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.

17.25.1   Detailed Description

This file contains the fundamental definitions that classes of the MTK rely on to be implemented. Examples of these definitions are the definition of fundamental data types, and global variables affecting the construction of mimetic operators, among others.

Author

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

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

Todo Test selective precision mechanism.

Definition in file mtk_roots.h.

17.26   mtk_roots.h

```
00001
00017 /*
00018 Copyright (C) 2015, Computational Science Research Center, San Diego State
00019 University. All rights reserved.
00020
00021 Redistribution and use in source and binary forms, with or without modification,
00022 are permitted provided that the following conditions are met:
00023
00024 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00025 and a copy of the modified files should be reported once modifications are
00026 completed. Documentation related to said modifications should be included.
00027
00028 2. Redistributions of source code must be done through direct
00029 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00030
00031 3. Redistributions of source code must retain the above copyright notice, this
00032 list of conditions and the following disclaimer.
00033
00034 4. Redistributions in binary form must reproduce the above copyright notice,
00035 this list of conditions and the following disclaimer in the documentation and/or
00036 other materials provided with the distribution.
00037
00038 5. Usage of the binary form on proprietary applications shall require explicit
00039 prior written permission from the the copyright holders.
00040
00041 6. Neither the name of the copyright holder nor the names of its contributors
00042 may be used to endorse or promote products derived from this software without
00043 specific prior written permission.
00044
00045 The copyright holders provide no reassurances that the source code provided does
00046 not infringe any patent, copyright, or any other intellectual property rights of
00047 third parties. The copyright holders disclaim any liability to any recipient for
00048 claims brought against recipient by any third party for infringement of that
00049 parties intellectual property rights.
00050
00051 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00052 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00053 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00054 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00055 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00056 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00057 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00058 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00059 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
```

```
00060 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00061 */
00062
00063 #ifndef MTK_INCLUDE_ROOTS_H_
00064 #define MTK_INCLUDE_ROOTS_H_
00065
00071 namespace mtk {
00072
00080 #ifdef MTK_PRECISION_DOUBLE
00081 typedef double Real;
00082 #else
00083 typedef float Real;
00084 #endif
00085
00103 #ifdef MTK_PRECISION_DOUBLE
00104 const double kZero{0.0};
00105 const double kOne{1.0};
00106 #else
00107 const float kZero{0.0f};
00108 const float kOne{1.0f};
00109 #endif
00110
00118 #ifdef MTK_PRECISION_DOUBLE
00119 const double kDefaultTolerance{1e-7};
00120 #else
00121 const float kDefaultTolerance{1e-7f};
00122 #endif
00123
00133 const int kDefaultOrderAccuracy{2};
00134
00144 #ifdef MTK_PRECISION_DOUBLE
00145 const double kDefaultMimeticThreshold{1.e-6};
00146 #else
00147 const float kDefaultMimeticThreshold{1.e-6f};
00148 #endif
00149
00157 const int kCriticalOrderAccuracyDiv{8};
00158
00166 const int kCriticalOrderAccuracyGrad{10};
00167 }
00168 #endif  // End of: MTK_INCLUDE_ROOTS_H_
```

17.27   include/mtk_tools.h File Reference

Tool manager class.

`#include <ctime>`
Include dependency graph for mtk_tools.h:
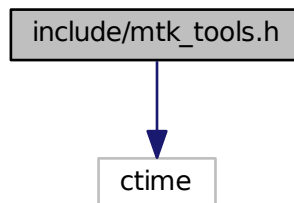
This graph shows which files directly or indirectly include this file:
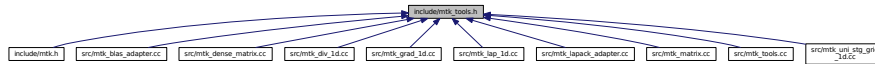
Classes

- class mtk::Tools

    Tool manager class.

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

17.27.1   Detailed Description

Basic tools to ensure execution correctness.

Author

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

Definition in file mtk_tools.h.

17.28   mtk_tools.h

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
```

```
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #ifndef MTK_INCLUDE_TOOLS_H_
00057 #define MTK_INCLUDE_TOOLS_H_
00058
00059 #include <ctime>
00060
00061 namespace mtk {
00062
00072 class Tools {
00073  public:
00084   static void Prevent(const bool condition,
00085                       const char *fname,
00086                       int lineno,
00087                       const char *fxname);
00088
00094   static void BeginTestNo(const int &nn);
00095
00101   static void EndTestNo(const int &nn);
00102
00103  private:
00104   static int test_number_;
00105
00106   static clock_t begin_time_;
00107 };
00108 }
00109 #endif  // End of: MTK_INCLUDE_TOOLS_H_
```

17.29 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:
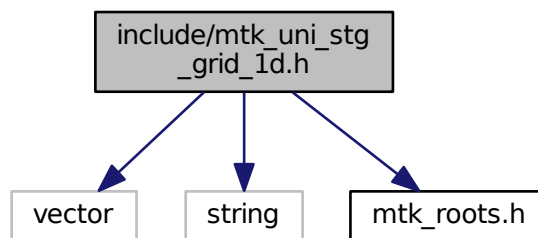
This graph shows which files directly or indirectly include this file:
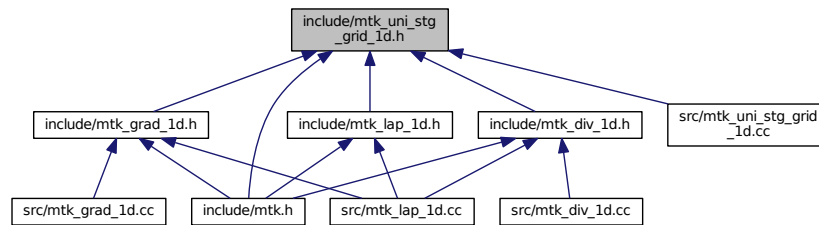
Classes

- class [mtk::UniStgGrid1D](#)

  Uniform 1D Staggered Grid.

Namespaces

- [mtk](#)

  Mimetic Methods Toolkit namespace.

17.29.1 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](#).

17.30 mtk_uni_stg_grid_1d.h

```
00001
00012 /*
00013 Copyright (C) 2015, Computational Science Research Center, San Diego State
00014 University. All rights reserved.
00015
00016 Redistribution and use in source and binary forms, with or without modification,
00017 are permitted provided that the following conditions are met:
00018
00019 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00020 and a copy of the modified files should be reported once modifications are
00021 completed. Documentation related to said modifications should be included.
00022
00023 2. Redistributions of source code must be done through direct
00024 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00025
00026 3. Redistributions of source code must retain the above copyright notice, this
00027 list of conditions and the following disclaimer.
```

```
00028
00029 4. Redistributions in binary form must reproduce the above copyright notice,
00030 this list of conditions and the following disclaimer in the documentation and/or
00031 other materials provided with the distribution.
00032
00033 5. Usage of the binary form on proprietary applications shall require explicit
00034 prior written permission from the the copyright holders.
00035
00036 6. Neither the name of the copyright holder nor the names of its contributors
00037 may be used to endorse or promote products derived from this software without
00038 specific prior written permission.
00039
00040 The copyright holders provide no reassurances that the source code provided does
00041 not infringe any patent, copyright, or any other intellectual property rights of
00042 third parties. The copyright holders disclaim any liability to any recipient for
00043 claims brought against recipient by any third party for infringement of that
00044 parties intellectual property rights.
00045
00046 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00047 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00048 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00049 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00050 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00051 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00052 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00053 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00054 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00055 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00056 */
00057
00058 #ifndef MTK_INCLUDE_UNI_STG_GRID_1D_H_
00059 #define MTK_INCLUDE_UNI_STG_GRID_1D_H_
00060
00061 #include <vector>
00062 #include <string>
00063
00064 #include "mtk_roots.h"
00065
00066 namespace mtk {
00067
00077 class UniStgGrid1D {
00078  public:
00080   friend std::ostream& operator <<(std::ostream& stream, UniStgGrid1D &in);
00081
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 delta_x() const;
00116
00122   Real *discrete_domain_x();
00123
00129   Real *discrete_field_u();
00130
00136   int num_cells_x() const;
00137
00143   void BindScalarField(Real (*ScalarField)(Real xx));
00144
00156   void BindVectorField(Real (*VectorField)(Real xx));
00157
00169   bool WriteToFile(std::string filename,
00170                    std::string space_name,
00171                    std::string field_name);
00172
00173  private:
00174   FieldNature nature_;
00175
00176   std::vector<Real> discrete_domain_x_;
00177   std::vector<Real> discrete_field_u_;
00178
00179   Real west_bndy_x_;
00180   Real east_bndy_x_;
00181   Real num_cells_x_;
00182   Real delta_x_;
```

```
00183 };
00184 }
00185 #endif  // End of: MTK_INCLUDE_UNI_STG_GRID_1D_H_
```

17.31   Makefile.inc File Reference

17.32   Makefile.inc

```
00001 # Makefile setup file for MTK.
00002
00003 SHELL := /bin/bash
00004
00005 # Please set the following variables up:
00006
00007 #   2. Absolute path to base directory of the MTK... where is the MTK?
00008 # _____
00009
00010 BASE = /home/ejspeiro/Dropbox/MTK
00011
00012 #   2. The machine (platform) identifier and required precision.
00013 # _____
00014
00015 # Options are:
00016 # - LINUX: A LINUX box installation.
00017 # - OSX: Uses OS X optimized solvers.
00018
00019 PLAT = LINUX
00020
00021 # Options are:
00022 # - SINGLE: Use 4 B floating point numbers.
00023 # - DOUBLE: Use 8 B floating point numbers.
00024
00025 PRECISION = DOUBLE
00026
00027 #   3. Optimized solvers and operations by means of ATLAS in Linux?
00028 # _____
00029
00030 # If you have selected OSX in step 1, then you don't need to worry about this.
00031
00032 # Options are ON xor OFF:
00033
00034 ATL_OPT = OFF
00035
00036 #   4. Paths to dependencies (header files for compiling).
00037 # _____
00038
00039 # GLPK include path (soon to go):
00040
00041 GLPK_INC = $(HOME)/Libraries/glpk-4.55/include
00042
00043 # Linux: If ATLAS optimization is ON, users should only provide the path to
00044 # ATLAS:
00045
00046 ATLAS_INC = $(HOME)/Libraries/ATLAS_3.8.4-CORE/include
00047
00048 # OS X: Do nothing.
00049
00050 #   5. Paths to dependencies (archive files for (static) linking).
00051 # _____
00052
00053 # GLPK linking path (soon to go):
00054
00055 GLPK_LIB = $(HOME)/Libraries/glpk-4.55/lib/libglpk.a
00056
00057 # If optimization is OFF, then provide the paths for:
00058
00059 BLAS_LIB = $(HOME)/Libraries/BLAS/libblas.a
00060 LAPACK_LIB = $(HOME)/Libraries/lapack-3.4.1/liblapack.a
00061
00062 # WARNING: Vendor libraries should be used whenever they are available.
00063
00064 # However, if optimization is ON, please provide the path the ATLAS' archive:
00065
00066 ATLAS_LIB   = $(HOME)/Libraries/ATLAS_3.8.4-CORE/ATLAS_3.8.4-BUILD-Citadel/lib
00067
```

```
00068 #   6. Compiler and its flags.
00069 #   _____
00070
00071 CC = g++
00072
00073 # Debug Level. Options are:
00074 # 0. NO debug at all NOR any run-time checks... be cautious!
00075 # 1. Verbose (execution messages) AND run-time checks.
00076 # 2. Level 1 plus intermediate scalar-valued results.
00077 # 3. Level 2 plus intermediate array-valued results.
00078
00079 DEBUG_LEVEL = 3
00080
00081 # Flags recommended for release code:
00082
00083 CCFLAGS = -Wall -O2
00084
00085 # Flags recommended for debugging code:
00086
00087 CCFLAGS = -Wall -g
00088
00089 #   7. Archiver, its flags, and ranlib:
00090 #   _____
00091
00092 ARCH      = ar
00093 ARCHFLAGS = cr
00094
00095 # If your system does not have "ranlib" then set: "RANLIB = echo":
00096
00097 RANLIB = echo
00098
00099 # But, if possible:
00100
00101 RANLIB = ranlib
00102
00103 #   8. Valgrind's memcheck options:
00104 #   _____
00105
00106 MEMCHECK_OPTS = -v --tool=memcheck --leak-check=full --show-leak-kinds=all \
00107 --track-origins=yes --freelist-vol=20000000
00108
00109 # Done!
00110
00111 #   _____
00112 #   _____
00113 #   _____
00114
00115 #   MTK-related.
00116 #   _____
00117
00118 SRC       = $(BASE)/src
00119 INCLUDE   = $(BASE)/include
00120 LIB       = $(BASE)/lib
00121 MTK_LIB   = $(LIB)/libmtk.a
00122 TESTS     = $(BASE)/tests
00123 EXAMPLES  = $(BASE)/examples
00124
00125 #   Compiling-related.
00126 #   _____
00127
00128 CCFLAGS += -std=c++11 -fPIC -DMTK_DEBUG_LEVEL=$(DEBUG_LEVEL) -I$(INCLUDE)  -c
00129
00130 ifeq ($(PRECISION),DOUBLE)
00131   CCFLAGS += -DMTK_PRECISION_DOUBLE
00132 else
00133   CCFLAGS += -DMTK_PRECISION_SINGLE
00134 endif
00135
00136 # Only the GLPK is included because the other dependencies are coded in Fortran.
00137
00138 ifeq ($(ATL_OPT),ON)
00139   CCFLAGS  += -I$(GLPK_INC) $(ATLAS_INC)
00140 else
00141   CCFLAGS  += -I$(GLPK_INC)
00142 endif
00143
00144 #   Linking-related.
00145 #   _____
00146
00147 NOOPT_LIBS  = $(LAPACK_LIB) $(BLAS_LIB) -lm $(GLPK_LIB) -lstdc++
00148
```

```
00149 OPT_LIBS    = -L$(ATLAS_LIB) -latlas -llapack -lblas -lm -latlas -lstdc++
00150
00151 ifeq ($(PLAT),OSX)
00152   LINKER  = g++
00153   LINKER  += -framework Accelerate $(GLPK_LIB) $(MTK_LIB)
00154 else
00155   ifeq ($(ATL_OPT),ON)
00156     LINKER  = g++
00157     LIBS = $(MTK_LIB)
00158     LIBS += $(OPT_LIBS)
00159   else
00160     LINKER  = gfortran
00161     LIBS = $(MTK_LIB)
00162     LIBS += $(NOOPT_LIBS)
00163   endif
00164 endif
00165
00166 #   Documentation-related.
00167 #   _____
00168
00169 DOCGEN      = doxygen
00170 DOCFILENAME = doc_config.dxcf
00171 DOC         = $(BASE)/doc
00172 DOCFILE     = $(BASE)/$(DOCFILENAME)
```

17.33   README.md File Reference

17.34   README.md

```
00001 # README File for the Mimetic Methods Toolkit (MTK)
00002
00003 By: **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu**
00004     _____
00005
00006 ## 1. Description
00007
00008 We define numerical methods that are based on discretizations preserving the
00009 properties of their continuum counterparts to be **mimetic**.
00010
00011 The **Mimetic Methods Toolkit (MTK)** is a C++ library for mimetic numerical
00012 methods. It is arranged as a set of classes for **mimetic quadratures**,
00013 **mimetic interpolation**, and **mimetic discretization** methods for the
00014 numerical solution of ordinary and partial differential equations.
00015     _____
00016
00017 ## 2. Dependencies
00018
00019 This README assumes all of these dependencies are installed in the following
00020 folder:
00021
00022 ```
00023 $(HOME)/Libraries/
00024 ```
00025
00026 In this version, the MTK optionally uses ATLAS-optimized BLAS and LAPACK
00027 routines for the internal computation on some of the layers. However, ATLAS
00028 requires both BLAS and LAPACK in order to create their optimized distributions.
00029 Therefore, the following dependencies tree arises:
00030
00031 ### For Linux:
00032
00033 1. LAPACK - Available from: http://www.netlib.org/lapack/
00034   1. BLAS - Available from: http://www.netlib.org/blas/
00035
00036 2. (Optional) ATLAS - Available from: http://math-atlas.sourceforge.net/
00037   1. BLAS - Available from: http://www.netlib.org/blas/
00038   2. LAPACK - Available from: http://www.netlib.org/lapack/
00039
00040 3. (Optional) Valgrind - Available from: http://valgrind.org/
00041
00042 4. (Optional) Doxygen - Available from http://www.stack.nl/~dimitri/doxygen/
00043
00044 ### For OS X:
00045
00046 There are no dependences for OS X.
```

```
00047      _____
00048
00049 ## 3. Installation
00050
00051 ### PART 1. CONFIGURATION OF THE MAKEFILE.
00052
00053 The following steps are required the build and test the MTK. Please use the
00054 accompanying `Makefile.inc` file, which should provide a solid template to
00055 start with. The following command provides help on the options for make:
00056
00057 ```
00058 $ make help
00059 -----
00060 Makefile for the MTK.
00061
00062 Options are:
00063 - make: builds only the library and the examples.
00064 - all: builds the library, the examples and the documentation.
00065 - mtklib: builds the library, i.e. generates the archive files.
00066 - tests: generates the tests.
00067 - examples: generates the examples.
00068 - gendoc: generates the documentation for the library.
00069 - checkheaders: checks syntax of the header files.
00070
00071 - clean: cleans ALL the generated files.
00072 - cleanlib: cleans the generated archive and object files.
00073 - cleantests: cleans the generated tests executables.
00074 - cleanexamples: 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 Examples and tests will also be built.
00091      _____
00092
00093 ## 4. Frequently Asked Questions
00094
00095 Q: Why haven't you guys implemented GBS to build the library?
00096 A: I'm on it as we speak! ;)
00097
00098 Q: When will the other flavors be ready?
00099 A: Soon! I'm working on getting help on developing those.
00100
00101 Q: Is there any main reference when it comes to the theory on Mimetic Methods?
00102 A: Yes! Check: http://www.csrc.sdsu.edu/mimetic-book
00103
00104 Q: Do I need to generate the documentation myself?
00105 A: You can if you want to... but if you DO NOT want to, just go to our website.
00106      _____
00107
00108 ## 5. Contact, Support, and Credits
00109
00110 The MTK is developed by researchers and adjuncts to the
00111 [Computational Science Research Center (CSRC)](http://www.csrc.sdsu.edu/)
00112 at [San Diego State University (SDSU)](http://www.sdsu.edu/).
00113
00114 Developers are members of:
00115
00116 1. Mimetic Numerical Methods Research and Development Group.
00117 2. Computational Geoscience Research and Development Group.
00118 3. Ocean Modeling Research and Development Group.
00119
00120 Currently the developers are:
00121
00122 - **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu** - @ejspeiro
00123 - Jose E. Castillo, Ph.D. - jcastillo at mail dot sdsu dot edu
00124 - Guillermo F. Miranda, Ph.D. - unigrav at hotmail dot com
00125 - Christopher P. Paolini, Ph.D. - paolini at engineering dot sdsu dot edu
00126 - Angel Boada.
00127 - Johnny Corbino.
```

```
00128 - Raul Vargas-Navarro.
00129
00130 Finally, please feel free to contact me with suggestions or corrections:
00131
00132 **Eduardo J. Sanchez, Ph.D. - esanchez at mail dot sdsu dot edu** - @ejspeiro
00133
00134 Thanks and happy coding!
```

17.35 src/mtk_blas_adapter.cc File Reference

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "mtk_roots.h"
#include "mtk_tools.h"
#include "mtk_blas_adapter.h"
```
Include dependency graph for mtk_blas_adapter.cc:

Namespaces

- mtk
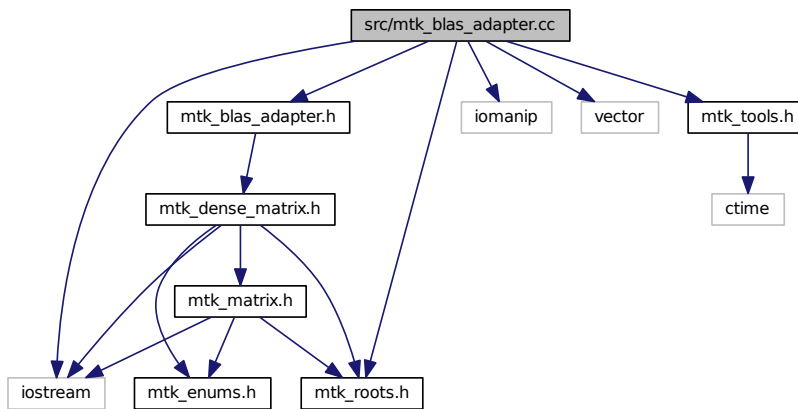
    Mimetic Methods Toolkit namespace.

Functions

- float mtk::snrm2_ (int *n, Real *x, int *incx)
- void mtk::sgemv_ (char *trans, int *m, int *n, Real *alpha, Real *a, int *lda, Real *x, int *incx, Real *beta, Real *y, int *incy)
- void mtk::sgemm_ (char *transa, char *transb, int *m, int *n, int *k, Real *alpha, Real *a, int *lda, Real *b, aamm int *ldb, Real *beta, Real *c, int *ldc)

17.36 mtk_blas_adapter.cc

```
00001
00024 /*
00025 Copyright (C) 2015, Computational Science Research Center, San Diego State
00026 University. All rights reserved.
00027
00028 Redistribution and use in source and binary forms, with or without modification,
00029 are permitted provided that the following conditions are met:
00030
00031 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00032 and a copy of the modified files should be reported once modifications are
00033 completed. Documentation related to said modifications should be included.
00034
00035 2. Redistributions of source code must be done through direct
00036 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00037
00038 3. Redistributions of source code must retain the above copyright notice, this
00039 list of conditions and the following disclaimer.
00040
00041 4. Redistributions in binary form must reproduce the above copyright notice,
00042 this list of conditions and the following disclaimer in the documentation and/or
00043 other materials provided with the distribution.
00044
00045 5. Usage of the binary form on proprietary applications shall require explicit
00046 prior written permission from the the copyright holders.
00047
00048 6. Neither the name of the copyright holder nor the names of its contributors
00049 may be used to endorse or promote products derived from this software without
00050 specific prior written permission.
00051
00052 The copyright holders provide no reassurances that the source code provided does
00053 not infringe any patent, copyright, or any other intellectual property rights of
00054 third parties. The copyright holders disclaim any liability to any recipient for
00055 claims brought against recipient by any third party for infringement of that
00056 parties intellectual property rights.
00057
00058 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00059 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00060 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00061 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00062 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00063 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00064 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00065 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00066 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00067 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00068 */
00069
00070 #include <iostream>
00071 #include <iomanip>
00072
00073 #include <vector>
00074
00075 #include "mtk_roots.h"
00076 #include "mtk_tools.h"
00077 #include "mtk_blas_adapter.h"
00078
00079 namespace mtk {
00080
00081 extern "C" {
00082
00083 #ifdef MTK_PRECISION_DOUBLE
00084
00095 double dnrm2_(int *n, Real *x, int *incx);
00096 #else
00097
00108 float snrm2_(int *n, Real *x, int *incx);
00109 #endif
00110
00111 #ifdef MTK_PRECISION_DOUBLE
00112
00140 void dgemv_(char *trans,
00141             int *m,
00142            int *n,
00143            Real *alpha,
00144            Real *a,
00145            int *lda,
00146            Real *x,
00147            int *incx,
```

```
00148               Real *beta,
00149               Real *y,
00150               int *incy);
00151 #else
00152
00180 void sgemv_(char *trans,
00181               int *m,
00182               int *n,
00183               Real *alpha,
00184               Real *a,
00185               int *lda,
00186               Real *x,
00187               int *incx,
00188               Real *beta,
00189               Real *y,
00190               int *incy);
00191 #endif
00192
00193 #ifdef MTK_PRECISION_DOUBLE
00194
00219 void dgemm_(char *transa,
00220               char* transb,
00221               int *m,
00222               int *n,
00223               int *k,
00224               Real *alpha,
00225               Real *a,
00226               int *lda,
00227               Real *b,
00228               int *ldb,
00229               Real *beta,
00230               Real *c,
00231               int *ldc);
00232 }
00233 #else
00234
00259 void sgemm_(char *transa,
00260               char* transb,
00261               int *m,
00262               int *n,
00263               int *k,
00264               Real *alpha,
00265               Real *a,
00266               int *lda,
00267               Real *b,aamm
00268               int *ldb,
00269               Real *beta,
00270               Real *c,
00271               int *ldc);
00272 }
00273 #endif
00274 }
00275
00276 mtk::Real mtk::BLASAdapter::RealNRM2(Real *in, int &in_length) {
00277
00278   int ix{1};  // Increment for the elements of xx. ix >= 0.
00279
00280   #ifdef MTK_PRECISION_DOUBLE
00281   return dnrm2_(&in_length, in, &ix);
00282   #else
00283   return snrm2_(&in_length, in, &ix);
00284   #endif
00285 }
00286
00287 void mtk::BLASAdapter::RealDenseMV(mtk::Real &alpha,
00288                                     mtk::DenseMatrix &aa,
00289                                     mtk::Real *xx,
00290                                     mtk::Real &beta,
00291                                     mtk::Real *yy) {
00292
00293   // Make sure input matrices are row-major ordered.
00294
00295   if (aa.matrix_properties().ordering() ==
00296     mtk::COL_MAJOR) {
00296     aa.OrderRowMajor();
00297   }
00298
00299   char transa{'T'}; // State that now, the input WILL be in row-major ordering.
00300
00301   int mm{aa.num_rows()};                  // Rows of aa.
00302   int nn{aa.num_cols()};                  // Columns of aa.
```

```
00303    int lda{(aa.matrix_properties()).ld()}; // Leading dimension.
00304    int incx{1};                                  // Increment of values in x.
00305    int incy{1};                                  // Increment of values in y.
00306
00307    std::swap(mm,nn);
00308    #ifdef MTK_PRECISION_DOUBLE
00309    dgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00310           xx, &incx, &beta, yy, &incy);
00311    #else
00312    sgemv_(&transa, &mm, &nn, &alpha, aa.data(), &lda,
00313           xx, &incx, &beta, yy, &incy);
00314    #endif
00315    std::swap(mm,nn);
00316 }
00317
00318 mtk::DenseMatrix mtk::BLASAdapter::RealDenseMM(
       mtk::DenseMatrix &aa,
00319                                                     mtk::DenseMatrix &bb) {
00320
00321    #if MTK_DEBUG_LEVEL > 0
00322    mtk::Tools::Prevent(aa.num_cols() != bb.num_rows(),
00323                        __FILE__, __LINE__, __func__);
00324    #endif
00325
00326    // Make sure input matrices are row-major ordered.
00327
00328    if (aa.matrix_properties().ordering() ==
       mtk::COL_MAJOR) {
00329       aa.OrderRowMajor();
00330    }
00331    if (bb.matrix_properties().ordering() ==
       mtk::COL_MAJOR) {
00332       bb.OrderRowMajor();
00333    }
00334
00335    char ta{'T'}; // State that input matrix aa is in row-wise ordering.
00336    char tb{'T'}; // State that input matrix bb is in row-wise ordering.
00337
00338    int mm{aa.num_rows()};  // Rows of aa and rows of cc.
00339    int nn{bb.num_cols()};  // Cols of bb and cols of cc.
00340    int kk{aa.num_cols()};  // Cols of aa and rows of bb.
00341
00342    int cc_num_rows{mm};  // Rows of cc.
00343    int cc_num_cols{nn};  // Columns of cc.
00344
00345    int lda{std::max(1,kk)};  // Leading dimension of the aa matrix.
00346    int ldb{std::max(1,nn)};  // Leading dimension of the bb matrix.
00347    int ldc{std::max(1,mm)};  // Leading dimension of the cc matrix.
00348
00349    mtk::Real alpha{1.0}; // First scalar coefficient.
00350    mtk::Real beta{0.0};  // Second scalar coefficient.
00351
00352    mtk::DenseMatrix cc_col_maj_ord(cc_num_rows,cc_num_cols); // Output matrix.
00353
00354    cc_col_maj_ord.SetOrdering(mtk::COL_MAJOR);
00355
00356    #ifdef MTK_PRECISION_DOUBLE
00357    dgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00358           bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00359    #else
00360    sgemm_(&ta, &tb, &mm, &nn, &kk, &alpha, aa.data(), &lda,
00361           bb.data(), &ldb, &beta, cc_col_maj_ord.data(), &ldc);
00362    #endif
00363
00364    #if MTK_DEBUG_LEVEL > 0
00365    std::cout << "cc_col_maj_ord =" << std::endl;
00366    std::cout << cc_col_maj_ord << std::endl;
00367    #endif
00368
00369    cc_col_maj_ord.OrderRowMajor();
00370
00371    return cc_col_maj_ord;
00372 }
```

17.37 src/mtk_dense_matrix.cc File Reference

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

```
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <typeinfo>
#include <algorithm>
#include "mtk_roots.h"
#include "mtk_dense_matrix.h"
#include "mtk_tools.h"
```
Include dependency graph for mtk_dense_matrix.cc:
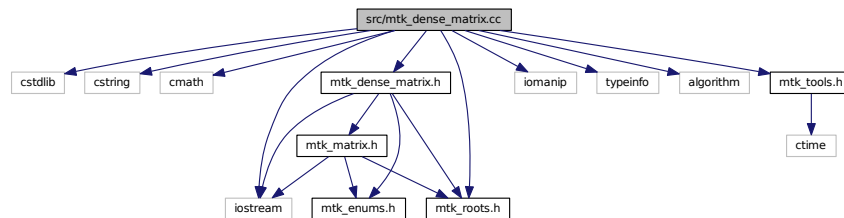
Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

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

17.37.1 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

Definition in file mtk_dense_matrix.cc.

17.38 mtk_dense_matrix.cc

```
00001
```

```
00013 /*
00014 Copyright (C) 2015, Computational Science Research Center, San Diego State
00015 University. All rights reserved.
00016
00017 Redistribution and use in source and binary forms, with or without modification,
00018 are permitted provided that the following conditions are met:
00019
00020 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00021 and a copy of the modified files should be reported once modifications are
00022 completed. Documentation related to said modifications should be included.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00026
00027 3. Redistributions of source code must retain the above copyright notice, this
00028 list of conditions and the following disclaimer.
00029
00030 4. Redistributions in binary form must reproduce the above copyright notice,
00031 this list of conditions and the following disclaimer in the documentation and/or
00032 other materials provided with the distribution.
00033
00034 5. Usage of the binary form on proprietary applications shall require explicit
00035 prior written permission from the the copyright holders.
00036
00037 6. Neither the name of the copyright holder nor the names of its contributors
00038 may be used to endorse or promote products derived from this software without
00039 specific prior written permission.
00040
00041 The copyright holders provide no reassurances that the source code provided does
00042 not infringe any patent, copyright, or any other intellectual property rights of
00043 third parties. The copyright holders disclaim any liability to any recipient for
00044 claims brought against recipient by any third party for infringement of that
00045 parties intellectual property rights.
00046
00047 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00048 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00049 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00050 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00051 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00052 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00053 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00054 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00055 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00056 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00057 */
00058
00059 #include <cstdlib>
00060 #include <cstring>
00061 #include <cmath>
00062
00063 #include <iostream>
00064 #include <iomanip>
00065 #include <typeinfo>
00066
00067 #include <algorithm>
00068
00069 #include "mtk_roots.h"
00070 #include "mtk_dense_matrix.h"
00071 #include "mtk_tools.h"
00072
00073 namespace mtk {
00074
00075 std::ostream& operator <<(std::ostream &stream, mtk::DenseMatrix &in) {
00076
00077   int mm{in.matrix_properties_.num_rows()};  // Auxiliary.
00078   int nn{in.matrix_properties_.num_cols()};  // Auxiliary.
00079
00080   if (in.matrix_properties_.ordering() ==
00081   mtk::COL_MAJOR) {
00081     std::swap(mm, nn);
00082   }
00083   for (auto ii = 0; ii < mm; ii++) {
00084     for (auto jj = 0; jj < nn; jj++) {
00085       mtk::Real value = in.data_[ii*nn + jj];
00086       stream << std::setw(13) << value;
00087     }
00088     stream << std::endl;
00089   }
00090   if (in.matrix_properties_.ordering() ==
00090   mtk::COL_MAJOR) {
00091     std::swap(mm, nn);
```

```
00092   }
00093   return stream;
00094 }
00095 }
00096
00097 mtk::DenseMatrix& mtk::DenseMatrix::operator =(const
      mtk::DenseMatrix &in) {
00098
00099   if(this == &in) {
00100     return *this;
00101   }
00102
00103   matrix_properties_.set_storage(in.
      matrix_properties_.storage());
00104
00105   matrix_properties_.set_ordering(in.
      matrix_properties_.ordering());
00106
00107   auto aux = in.matrix_properties_.num_rows();
00108   matrix_properties_.set_num_rows(aux);
00109
00110   aux = in.matrix_properties().num_cols();
00111   matrix_properties_.set_num_cols(aux);
00112
00113   aux = in.matrix_properties().num_zero();
00114   matrix_properties_.set_num_zero(aux);
00115
00116   aux = in.matrix_properties().num_null();
00117   matrix_properties_.set_num_null(aux);
00118
00119   auto num_rows = matrix_properties_.num_rows();
00120   auto num_cols = matrix_properties_.num_cols();
00121
00122   delete [] data_;
00123
00124   try {
00125     data_ = new mtk::Real[num_rows*num_cols];
00126   } catch (std::bad_alloc &memory_allocation_exception) {
00127     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00128       std::endl;
00129     std::cerr << memory_allocation_exception.what() << std::endl;
00130   }
00131   memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*
      num_cols);
00132
00133   std::copy(in.data_, in.data_ + num_rows*num_cols, data_);
00134
00135   return *this;
00136 }
00137
00138 mtk::DenseMatrix::DenseMatrix(): data_(nullptr) {
00139
00140   matrix_properties_.set_storage(mtk::DENSE);
00141   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00142 }
00143
00144 mtk::DenseMatrix::DenseMatrix(const
      mtk::DenseMatrix &in) {
00145
00146   matrix_properties_.set_storage(in.matrix_properties_.storage());
00147
00148   matrix_properties_.set_ordering(in.matrix_properties_.
      ordering());
00149
00150   auto aux = in.matrix_properties_.num_rows();
00151   matrix_properties_.set_num_rows(aux);
00152
00153   aux = in.matrix_properties().num_cols();
00154   matrix_properties_.set_num_cols(aux);
00155
00156   aux = in.matrix_properties().num_zero();
00157   matrix_properties_.set_num_zero(aux);
00158
00159   aux = in.matrix_properties().num_null();
00160   matrix_properties_.set_num_null(aux);
00161
00162   auto num_rows = in.matrix_properties_.num_rows();
00163   auto num_cols = in.matrix_properties_.num_cols();
00164
00165   try {
00166     data_ = new mtk::Real[num_rows*num_cols];
```

```
00167   } catch (std::bad_alloc &memory_allocation_exception) {
00168     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00169       std::endl;
00170     std::cerr << memory_allocation_exception.what() << std::endl;
00171   }
00172   memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00173
00174   std::copy(in.data_,in.data_ + num_rows*num_cols,data_);
00175 }
00176
00177 mtk::DenseMatrix::DenseMatrix(const int &num_rows, const int &num_cols) {
00178
00179   #if MTK_DEBUG_LEVEL > 0
00180   mtk::Tools::Prevent(num_rows < 1, __FILE__, __LINE__, __func__);
00181   mtk::Tools::Prevent(num_cols < 1, __FILE__, __LINE__, __func__);
00182   #endif
00183
00184   matrix_properties_.set_storage(mtk::DENSE);
00185   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00186   matrix_properties_.set_num_rows(num_rows);
00187   matrix_properties_.set_num_cols(num_cols);
00188
00189   try {
00190     data_ = new mtk::Real[num_rows*num_cols];
00191   } catch (std::bad_alloc &memory_allocation_exception) {
00192     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00193       std::endl;
00194     std::cerr << memory_allocation_exception.what() << std::endl;
00195   }
00196   memset(data_, mtk::kZero, sizeof(data_[0])*num_rows*num_cols);
00197 }
00198
00199 mtk::DenseMatrix::DenseMatrix(const int &rank,
00200                              const bool &padded,
00201                              const bool &transpose) {
00202
00203   #if MTK_DEBUG_LEVEL > 0
00204   mtk::Tools::Prevent(rank < 1, __FILE__, __LINE__, __func__);
00205   #endif
00206
00207   int aux{};  // Used to control the padding.
00208
00209   if (padded) {
00210     aux = 1;
00211   }
00212
00213   matrix_properties_.set_storage(mtk::DENSE);
00214   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00215   matrix_properties_.set_num_rows(aux + rank + aux);
00216   matrix_properties_.set_num_cols(rank);
00217
00218   try {
00219     data_ = new mtk::Real[matrix_properties_.num_values()];
00220   } catch (std::bad_alloc &memory_allocation_exception) {
00221     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00222       std::endl;
00223     std::cerr << memory_allocation_exception.what() << std::endl;
00224   }
00225   memset(data_,
00226          mtk::kZero,
00227          sizeof(data_[0])*(matrix_properties_.num_values()));
00228
00229   for (auto ii =0 ; ii < matrix_properties_.num_rows(); ++ii) {
00230     for (auto jj = 0; jj < matrix_properties_.num_cols(); ++jj) {
00231       data_[ii*matrix_properties_.num_cols() + jj] =
00232         (ii == jj + aux)? mtk::kOne: mtk::kZero;
00233     }
00234   }
00235 }
00236
00237 mtk::DenseMatrix::DenseMatrix(const mtk::Real *gen,
00238                              const int &gen_length,
00239                              const int &pro_length,
00240                              const bool &transpose) {
00241
00242   #if MTK_DEBUG_LEVEL > 0
00243   mtk::Tools::Prevent(gen == nullptr, __FILE__, __LINE__, __func__);
00244   mtk::Tools::Prevent(gen_length < 1, __FILE__, __LINE__, __func__);
00245   mtk::Tools::Prevent(pro_length < 1, __FILE__, __LINE__, __func__);
00246   #endif
00247
```

```
00248   matrix_properties_.set_storage(mtk::DENSE);
00249   matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00250   if (!transpose) {
00251     matrix_properties_.set_num_rows(gen_length);
00252     matrix_properties_.set_num_cols(pro_length);
00253   } else {
00254     matrix_properties_.set_num_rows(pro_length);
00255     matrix_properties_.set_num_cols(gen_length);
00256   }
00257
00258   int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00259   int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00260
00261   try {
00262     data_ = new mtk::Real[rr*cc];
00263   } catch (std::bad_alloc &memory_allocation_exception) {
00264     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00265       std::endl;
00266     std::cerr << memory_allocation_exception.what() << std::endl;
00267   }
00268   memset(data_, mtk::kZero, sizeof(data_[0])*rr*cc);
00269
00270   if (!transpose) {
00271     for (auto ii = 0; ii < rr; ii++) {
00272       for (auto jj = 0; jj < cc; jj++) {
00273         data_[ii*cc + jj] = pow(gen[ii], (double) jj);
00274       }
00275     }
00276   } else {
00277     for (auto ii = 0; ii < rr; ii++) {
00278       for (auto jj = 0; jj < cc; jj++) {
00279         data_[ii*cc + jj] = pow(gen[jj], (double) ii);
00280       }
00281     }
00282   }
00283 }
00284
00285 mtk::DenseMatrix::~DenseMatrix() {
00286
00287   delete[] data_;
00288   data_ = nullptr;
00289 }
00290
00291 mtk::Matrix mtk::DenseMatrix::matrix_properties() const {
00292
00293   return matrix_properties_;
00294 }
00295
00296 void mtk::DenseMatrix::SetOrdering(
00297     mtk::MatrixOrdering oo) {
00298   #if MTK_DEBUG_LEVEL > 0
00299   mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
00300   mtk::COL_MAJOR),
00300                       __FILE__, __LINE__, __func__);
00301   #endif
00302
00303   matrix_properties_.set_ordering(oo);
00304 }
00305
00306 int mtk::DenseMatrix::num_rows() const {
00307
00308   return matrix_properties_.num_rows();
00309 }
00310
00311 int mtk::DenseMatrix::num_cols() const {
00312
00313   return matrix_properties_.num_cols();
00314 }
00315
00316 mtk::Real* mtk::DenseMatrix::data() const {
00317
00318   return data_;
00319 }
00320
00321 mtk::Real mtk::DenseMatrix::GetValue(
00322     const int &rr,
00323     const int &cc) const {
00324
00325   #if MTK_DEBUG_LEVEL > 0
00326   mtk::Tools::Prevent(rr < 0, __FILE__, __LINE__, __func__);
```

```
00327    mtk::Tools::Prevent(cc < 0, __FILE__, __LINE__, __func__);
00328    #endif
00329
00330    return data_[rr*matrix_properties_.num_cols() + cc];
00331  }
00332
00333  void  mtk::DenseMatrix::SetValue(
00334      const int &rr,
00335      const int &cc,
00336      const mtk::Real &val) {
00337
00338    #if MTK_DEBUG_LEVEL > 0
00339    mtk::Tools::Prevent(rr < 0, __FILE__, __LINE__, __func__);
00340    mtk::Tools::Prevent(cc < 0, __FILE__, __LINE__, __func__);
00341    #endif
00342
00343    data_[rr*matrix_properties_.num_cols() + cc] = val;
00344  }
00345
00346  void mtk::DenseMatrix::Transpose() {
00347
00349
00350    mtk::Real *data_transposed{}; // Buffer.
00351
00352    int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00353    int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00354
00355    try {
00356      data_transposed = new mtk::Real[rr*cc];
00357    } catch (std::bad_alloc &memory_allocation_exception) {
00358      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00359        std::endl;
00360      std::cerr << memory_allocation_exception.what() << std::endl;
00361    }
00362    memset(data_transposed,
00363           mtk::kZero,
00364           sizeof(data_transposed[0])*rr*cc);
00365
00366    // Assign the values to their transposed position.
00367    for (auto ii = 0; ii < rr; ++ii) {
00368      for (auto jj = 0; jj < cc; ++jj) {
00369        data_transposed[jj*rr + ii] = data_[ii*cc + jj];
00370      }
00371    }
00372
00373    // Swap pointers.
00374    auto tmp = data_; // Temporal holder.
00375    data_ = data_transposed;
00376    delete [] tmp;
00377    tmp = nullptr;
00378
00379    matrix_properties_.set_num_rows(cc);
00380    matrix_properties_.set_num_cols(rr);
00381  }
00382
00383  void mtk::DenseMatrix::OrderRowMajor() {
00384
00385    if (matrix_properties_.ordering() == mtk::COL_MAJOR) {
00386
00388
00389      mtk::Real *data_transposed{}; // Buffer.
00390
00391      int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00392      int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00393
00394      try {
00395        data_transposed = new mtk::Real[rr*cc];
00396      } catch (std::bad_alloc &memory_allocation_exception) {
00397        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00398          std::endl;
00399        std::cerr << memory_allocation_exception.what() << std::endl;
00400      }
00401      memset(data_transposed,
00402             mtk::kZero,
00403             sizeof(data_transposed[0])*rr*cc);
00404
00405      // Assign the values to their transposed position.
00406      std::swap(rr, cc);
00407      for (auto ii = 0; ii < rr; ++ii) {
00408        for (auto jj = 0; jj < cc; ++jj) {
00409          data_transposed[jj*rr + ii] = data_[ii*cc + jj];
```

```
00410        }
00411      }
00412      std::swap(rr, cc);
00413
00414      // Swap pointers.
00415      auto tmp = data_; // Temporal holder.
00416      data_ = data_transposed;
00417      delete [] tmp;
00418      tmp = nullptr;
00419
00420      matrix_properties_.set_ordering(mtk::ROW_MAJOR);
00421    }
00422  }
00423
00424  void mtk::DenseMatrix::OrderColMajor() {
00425
00426    if (matrix_properties_.ordering() == ROW_MAJOR) {
00427
00429
00430      mtk::Real *data_transposed{}; // Buffer.
00431
00432      int rr = matrix_properties_.num_rows(); // Used to construct this matrix.
00433      int cc = matrix_properties_.num_cols(); // Used to construct this matrix.
00434
00435      try {
00436        data_transposed = new mtk::Real[rr*cc];
00437      } catch (std::bad_alloc &memory_allocation_exception) {
00438        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00439          std::endl;
00440        std::cerr << memory_allocation_exception.what() << std::endl;
00441      }
00442      memset(data_transposed,
00443             mtk::kZero,
00444             sizeof(data_transposed[0])*rr*cc);
00445
00446      // Assign the values to their transposed position.
00447      for (auto ii = 0; ii < rr; ++ii) {
00448        for (auto jj = 0; jj < cc; ++jj) {
00449          data_transposed[jj*rr + ii] = data_[ii*cc + jj];
00450        }
00451      }
00452
00453      // Swap pointers.
00454      auto tmp = data_; // Temporal holder.
00455      data_ = data_transposed;
00456      delete [] tmp;
00457      tmp = nullptr;
00458
00459      matrix_properties_.set_ordering(mtk::COL_MAJOR);
00460    }
00461  }
00462
00463  mtk::DenseMatrix mtk::DenseMatrix::Kron(const
       mtk::DenseMatrix &aa,
00464                                          const mtk::DenseMatrix &bb) {
00465
00466    int row_offset{}; // Offset for rows.
00467    int col_offset{}; // Offset for rows.
00468
00469    mtk::Real aa_factor{}; // Used in computation.
00470
00471    // Auxiliary variables:
00472    auto aux1 = aa.matrix_properties_.num_rows()*bb.
       matrix_properties_.num_rows();
00473    auto aux2 = aa.matrix_properties_.num_cols()*bb.
       matrix_properties_.num_cols();
00474
00475    mtk::DenseMatrix output(aux1,aux2); // Output matrix.
00476
00477    int kk_num_cols{output.matrix_properties_.num_cols()}; // Aux.
00478
00479    auto mm = aa.matrix_properties_.num_rows(); // Rows of aa.
00480    auto nn = aa.matrix_properties_.num_cols(); // Cols of aa.
00481    auto pp = bb.matrix_properties_.num_rows(); // Rows of bb.
00482    auto qq = bb.matrix_properties_.num_cols(); // Cols of bb.
00483
00484    for (auto ii = 0; ii < mm; ++ii) {
00485      row_offset = ii*pp;
00486      for (auto jj = 0; jj < nn; ++jj) {
00487        col_offset = jj*qq;
00488        aa_factor = aa.data_[ii*nn + jj];
```

```
00489        for (auto ll = 0; ll < pp; ++ll) {
00490          for (auto oo = 0; oo < qq; ++oo) {
00491            auto index = (ll + row_offset)*kk_num_cols + (oo + col_offset);
00492            output.data_[index] = aa_factor*bb.data_[ll*qq + oo];
00493          }
00494        }
00495      }
00496    }
00497
00498    output.matrix_properties_.set_storage(mtk::DENSE);
00499    output.matrix_properties_.set_ordering(
    mtk::ROW_MAJOR);
00500
00501    return output;
00502 }
```

17.39 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)

17.39.1 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.

17.40 mtk_div_1d.cc

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed. Documentation related to said modifications should be included.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066 #include <limits>
```

```
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074
00075 #include "mtk_div_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Div1D &in) {
00080
00081
00082
00083   stream << "divergence_[0] = " << std::setw(9) << in.divergence_[0] <<
00084     std::endl;
00085
00086
00087
00088   stream << "divergence_[1:" << in.order_accuracy_ << "] = ";
00089   for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00090     stream << std::setw(9) << in.divergence_[ii] << " ";
00091   }
00092   stream << std::endl;
00093
00094   if (in.order_accuracy_ > 2) {
00095
00096
00097
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
00105
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   #if MTK_DEBUG_LEVEL > 0
00180   mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00181   mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00182   mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00183                       __FILE__, __LINE__, __func__);
00184
00185   if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00186     std::cout << "WARNING: Numerical accuracy is critical." << std::endl;
00187   }
00188
00189   std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00190   std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00191   #endif
00192
00193   order_accuracy_ = order_accuracy;
00194   mimetic_threshold_ = mimetic_threshold;
00195
00197
00198   bool abort_construction = ComputeStencilInteriorGrid();
00199
00200   #if MTK_DEBUG_LEVEL > 0
00201   if (!abort_construction) {
00202     std::cerr << "Could NOT complete stage 1." << std::endl;
00203     std::cerr << "Exiting..." << std::endl;
00204     return false;
00205   }
00206   #endif
00207
00208   // At this point, we already have the values for the interior stencil stored
00209   // in the coeffs_interior_ array.
00210
00211   // It is noteworthy, that the 2nd-order-accurate divergence operator has NO
00212   // approximation at the boundary, thus it has no weights. For this case, the
00213   // dimension of the null-space of the Vandermonde matrices used to compute the
00214   // approximating coefficients at the boundary is 0. Ergo, we compute this
00215   // number first and then decide if we must compute anything at the boundary.
00216
00217   dim_null_ = order_accuracy_/2 - 1;
00218
00219   if (dim_null_ > 0) {
00220
00221     #ifdef MTK_PRECISION_DOUBLE
00222     num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00223     #else
00224     num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00225     #endif
00226
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     #if MTK_DEBUG_LEVEL > 0
00249     if (!abort_construction) {
00250       std::cerr << "Could NOT complete stage 2.1." << std::endl;
00251       std::cerr << "Exiting..." << std::endl;
00252       return false;
00253     }
00254     #endif
00255
00257
00258     abort_construction = ComputePreliminaryApproximations();
00259
00260     #if MTK_DEBUG_LEVEL > 0
00261     if (!abort_construction) {
00262       std::cerr << "Could NOT complete stage 2.2." << std::endl;
00263       std::cerr << "Exiting..." << std::endl;
00264       return false;
00265     }
00266     #endif
00267
00269
00270     abort_construction = ComputeWeights();
00271
00272     #if MTK_DEBUG_LEVEL > 0
00273     if (!abort_construction) {
00274       std::cerr << "Could NOT complete stage 2.3." << std::endl;
00275       std::cerr << "Exiting..." << std::endl;
00276       return false;
00277     }
00278     #endif
00279
00281
00282     abort_construction = ComputeStencilBoundaryGrid();
00283
00284     #if MTK_DEBUG_LEVEL > 0
00285     if (!abort_construction) {
00286       std::cerr << "Could NOT complete stage 2.4." << std::endl;
00287       std::cerr << "Exiting..." << std::endl;
00288       return false;
00289     }
00290     #endif
00291
00292   } // End of: if (dim_null_ > 0);
00293
00295
00296   // Once we have the following three collections of data:
00297   //   (a) the coefficients for the interior,
00298   //   (b) the coefficients for the boundary (if it applies),
00299   //   (c) and the weights (if it applies),
00300   // we will store everything in the output array:
00301
00302   abort_construction = AssembleOperator();
00303
00304   #if MTK_DEBUG_LEVEL > 0
00305   if (!abort_construction) {
00306     std::cerr << "Could NOT complete stage 3." << std::endl;
00307     std::cerr << "Exiting..." << std::endl;
00308     return false;
00309   }
00310   #endif
00311
00312   return true;
00313 }
00314
00315 mtk::Real *mtk::Div1D::weights_crs() {
00316
00317   return weights_crs_;
00318 }
```

```
00319
00320 mtk::Real *mtk::Div1D::weights_cbs() {
00321
00322   return weights_cbs_;
00323 }
00324
00325 mtk::DenseMatrix mtk::Div1D::ReturnAsDenseMatrix(const
      UniStgGrid1D &grid) {
00326
00327   int nn{grid.num_cells_x()}; // Number of cells on the grid.
00328
00329   #if MTK_DEBUG_LEVEL > 0
00330   mtk::Tools::Prevent(order_accuracy_ <= 0, __FILE__, __LINE__, __func__);
00331   mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00332   #endif
00333
00334   mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00335
00336   int dd_num_rows = nn + 2;
00337   int dd_num_cols = nn + 1;
00338   int elements_per_row = num_bndy_coeffs_;
00339   int num_extra_rows = dim_null_;
00340
00341   // Output matrix featuring sizes for divergence operators.
00342   mtk::DenseMatrix out(dd_num_rows, dd_num_cols);
00343
00345
00346   auto ee_index = 0;
00347   for (auto ii = 1; ii < num_extra_rows + 1; ii++) {
00348     auto cc = 0;
00349     for(auto jj = 0 ; jj < dd_num_rows; jj++) {
00350       if( cc >= elements_per_row) {
00351         out.SetValue(ii, jj, mtk::kZero);
00352       } else {
00353         out.SetValue(ii,jj, mim_bndy_[ee_index++]*inv_delta_x);
00354         cc++;
00355       }
00356     }
00357   }
00358
00360
00361   for (auto ii = num_extra_rows + 1;
00362        ii < dd_num_rows - num_extra_rows - 1; ii++) {
00363     auto jj = ii - num_extra_rows - 1;
00364     for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00365       out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00366     }
00367   }
00368
00370
00371   ee_index = 0;
00372   for (auto ii = dd_num_rows - 2; ii >= dd_num_rows - num_extra_rows - 1; ii--) {
00373     auto cc = 0;
00374     for (auto jj = dd_num_cols - 1; jj >= 0; jj--) {
00375       if( cc >= elements_per_row) {
00376         out.SetValue(ii,jj,0.0);
00377       } else {
00378         out.SetValue(ii,jj,-mim_bndy_[ee_index++]*inv_delta_x);
00379         cc++;
00380       }
00381     }
00382   }
00383
00384   return out;
00385 }
00386
00387 bool mtk::Div1D::ComputeStencilInteriorGrid() {
00388
00390
00391   mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00392
00393   try {
00394     pp = new mtk::Real[order_accuracy_];
00395   } catch (std::bad_alloc &memory_allocation_exception) {
00396     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00397       std::endl;
00398     std::cerr << memory_allocation_exception.what() << std::endl;
00399   }
00400   memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00401
00402   #ifdef MTK_PRECISION_DOUBLE
```

```
00403    pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00404    #else
00405    pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00406    #endif
00407
00408    for (auto ii = 1; ii < order_accuracy_; ++ii) {
00409      pp[ii] = pp[ii - 1] + mtk::kOne;
00410    }
00411
00412    #if MTK_DEBUG_LEVEL > 0
00413    std::cout << "pp =" << std::endl;
00414    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00415      std::cout << std::setw(12) << pp[ii];
00416    }
00417    std::cout << std::endl << std::endl;
00418    #endif
00419
00421    bool transpose{false};
00422
00423    mtk::DenseMatrix vander_matrix(pp,
00424                                   order_accuracy_,
00425                                   order_accuracy_,
00426                                   transpose);
00427
00428    #if MTK_DEBUG_LEVEL > 0
00429    std::cout << "vander_matrix = " << std::endl;
00430    std::cout << vander_matrix << std::endl;
00431    #endif
00432
00433
00435    try {
00436      coeffs_interior_ = new mtk::Real[order_accuracy_];
00437    } catch (std::bad_alloc &memory_allocation_exception) {
00438      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00439        std::endl;
00440      std::cerr << memory_allocation_exception.what() << std::endl;
00441    }
00442    memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00443
00445    coeffs_interior_[1] = mtk::kOne;
00446
00447    #if MTK_DEBUG_LEVEL > 0
00448    std::cout << "oo =" << std::endl;
00449    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00450      std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00451    }
00452    std::cout << std::endl;
00453    #endif
00454
00456
00457    int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00458                                                  coeffs_interior_)};
00459
00460    #if MTK_DEBUG_LEVEL > 0
00461    if (!info) {
00462      std::cout << "System solved! Interior stencil attained!" << std::endl;
00463      std::cout << std::endl;
00464    }
00465    else {
00466      std::cerr << "Something wrong solving system! info = " << info << std::endl;
00467      std::cerr << "Exiting..." << std::endl;
00468      return false;
00469    }
00470    #endif
00471
00472    #if MTK_DEBUG_LEVEL > 0
00473    std::cout << "coeffs_interior_ =" << std::endl;
00474    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00475      std::cout << std::setw(12) << coeffs_interior_[ii];
00476    }
00477    std::cout << std::endl << std::endl;
00478    #endif
00479
00480    delete [] pp;
00481    pp = nullptr;
00482
00483    return true;
00484 }
00485
00486 bool mtk::Div1D::ComputeRationalBasisNullSpace(void) {
```

```
00487
00488    mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00489
00490
00491
00492    try {
00493      gg = new mtk::Real[num_bndy_coeffs_];
00494    } catch (std::bad_alloc &memory_allocation_exception) {
00495      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00496        std::endl;
00497      std::cerr << memory_allocation_exception.what() << std::endl;
00498    }
00499    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00500
00501    #ifdef MTK_PRECISION_DOUBLE
00502    gg[0] = -1.0/2.0;
00503    #else
00504    gg[0] = -1.0f/2.0f;
00505    #endif
00506    for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00507      gg[ii] = gg[ii - 1] + mtk::kOne;
00508    }
00509
00510    #if MTK_DEBUG_LEVEL > 0
00511    std::cout << "gg =" << std::endl;
00512    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00513      std::cout << std::setw(12) << gg[ii];
00514    }
00515    std::cout << std::endl << std::endl;
00516    #endif
00517
00518
00519
00520    bool tran{true}; // Should I transpose the Vandermonde matrix.
00521
00522    mtk::DenseMatrix vv_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
00523
00524    #if MTK_DEBUG_LEVEL > 0
00525    std::cout << "vv_west_t =" << std::endl;
00526    std::cout << vv_west_t << std::endl;
00527    #endif
00528
00529
00530
00531    mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
       (vv_west_t));
00532
00533    #if MTK_DEBUG_LEVEL > 0
00534    std::cout << "QQ^T = " << std::endl;
00535    std::cout << qq_t << std::endl;
00536    #endif
00537
00538
00539
00540    int KK_num_rows_{num_bndy_coeffs_};
00541    int KK_num_cols_{dim_null_};
00542
00543    mtk::DenseMatrix KK(KK_num_rows_, KK_num_cols_);
00544
00545    for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00546      for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
00547        KK.data()[jj*dim_null_ + (ii - (num_bndy_coeffs_ - dim_null_))] =
00548          qq_t.data()[ii*num_bndy_coeffs_ + jj];
00549      }
00550    }
00551
00552    #if MTK_DEBUG_LEVEL > 0
00553    std::cout << "KK =" << std::endl;
00554    std::cout << KK << std::endl;
00555    std::cout << "KK.num_rows() = " << KK.num_rows() << std::endl;
00556    std::cout << "KK.num_cols() = " << KK.num_cols() << std::endl;
00557    std::cout << std::endl;
00558    #endif
00559
00560
00561
00562    // Scale thus requesting that the last entries of the attained basis for the
00563    // null-space, adopt the pattern we require.
00564    // Essentially we will implement the following MATLAB pseudo-code:
00565    //  scalers = KK(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00566    //  SK = KK*scalers
00567    // where SK is the scaled null-space.
00568
00569    // In this point, we almost have all the data we need correctly allocated
00570    // in memory. We will create the matrix II_, and elements we wish to scale in
00571    // the KK array. Using the concept of the leading dimension, we could just
```

```
00572    // use KK, with the correct leading dimension and that is it. BUT I DO NOT
00573    // GET how does it work. So I will just create a matrix with the content of
00574    // this array that we need, solve for the scalers and then scale the
00575    // whole KK:
00576
00577    // We will then create memory for that sub-matrix of KK (SUBK).
00578
00579    mtk::DenseMatrix SUBK(dim_null_,dim_null_);
00580
00581    for (auto ii = num_bndy_coeffs_ - dim_null_; ii < num_bndy_coeffs_; ++ii) {
00582      for (auto jj = 0; jj < dim_null_; ++jj) {
00583        SUBK.data()[(ii - (num_bndy_coeffs_ - dim_null_))*dim_null_ + jj] =
00584            KK.data()[ii*dim_null_ + jj];
00585      }
00586    }
00587
00588    #if MTK_DEBUG_LEVEL > 0
00589    std::cout << "SUBK =" << std::endl;
00590    std::cout << SUBK << std::endl;
00591    #endif
00592
00593    SUBK.Transpose();
00594
00595    #if MTK_DEBUG_LEVEL > 0
00596    std::cout << "SUBK^T =" << std::endl;
00597    std::cout << SUBK << std::endl;
00598    #endif
00599
00600    bool padded{false};
00601    tran = false;
00602
00603    mtk::DenseMatrix II(dim_null_, padded, tran);
00604
00605    #if MTK_DEBUG_LEVEL > 0
00606    std::cout << "II =" << std::endl;
00607    std::cout << II << std::endl;
00608    #endif
00609
00610    // Solve the system to compute the scalers.
00611    // An example of the system to solve, for k = 8, is:
00612    //
00613    // SUBK*scalers = II_ or
00614    //
00615    // |  0.386018  -0.0339244   -0.129478 |          | 1 0 0 |
00616    // | -0.119774   0.0199423   0.0558632 |*scalers = | 0 1 0 |
00617    // | 0.0155708 -0.00349546 -0.00853182 |          | 0 0 1 |
00618    //
00619    // Notice this is a nrhs = 3 system.
00620    // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00621    // will be stored in the created identity matrix.
00622    // Let us first transpose SUBK (because of LAPACK):
00623
00624    int info{mtk::LAPACKAdapter::SolveDenseSystem(SUBK, II)};
00625
00626    #if MTK_DEBUG_LEVEL > 0
00627    if (!info) {
00628      std::cout << "System successfully solved!" <<
00629        std::endl;
00630    } else {
00631      std::cerr << "Something went wrong solving system! info = " << info <<
00632        std::endl;
00633      std::cerr << "Exiting..." << std::endl;
00634      return false;
00635    }
00636    std::cout << std::endl;
00637    #endif
00638
00639    #if MTK_DEBUG_LEVEL > 0
00640    std::cout << "Computed scalers:" << std::endl;
00641    std::cout << II << std::endl;
00642    #endif
00643
00644    // Multiply the two matrices to attain a scaled basis for null-space.
00645
00646    rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(KK, II);
00647
00648    #if MTK_DEBUG_LEVEL > 0
00649    std::cout << "Rational basis for the null-space:" << std::endl;
00650    std::cout << rat_basis_null_space_ << std::endl;
00651    #endif
00652
```

```
00653    // At this point, we have a rational basis for the null-space, with the
00654    // pattern we need! :)
00655
00656    delete [] gg;
00657    gg = nullptr;
00658
00659    return true;
00660 }
00661
00662 bool mtk::Div1D::ComputePreliminaryApproximations(void) {
00663
00664
00665
00666    mtk::Real *gg{}; // Generator vector for the first approximation.
00667
00668    try {
00669      gg = new mtk::Real[num_bndy_coeffs_];
00670    } catch (std::bad_alloc &memory_allocation_exception) {
00671      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00672 std::endl;
00673      std::cerr << memory_allocation_exception.what() << std::endl;
00674    }
00675    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00676
00677    #ifdef MTK_PRECISION_DOUBLE
00678    gg[0] = -1.0/2.0;
00679    #else
00680    gg[0] = -1.0f/2.0f;
00681    #endif
00682    for (auto ii = 1; ii < num_bndy_coeffs_; ++ii) {
00683      gg[ii] = gg[ii - 1] + mtk::kOne;
00684    }
00685
00686    #if MTK_DEBUG_LEVEL > 0
00687    std::cout << "gg0 =" << std::endl;
00688    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00689      std::cout << std::setw(12) << gg[ii];
00690    }
00691    std::cout << std::endl << std::endl;
00692    #endif
00693
00694    // Allocate 2D array to store the collection of preliminary approximations.
00695    try {
00696      prem_apps_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
00697    } catch (std::bad_alloc &memory_allocation_exception) {
00698      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00699 std::endl;
00700      std::cerr << memory_allocation_exception.what() << std::endl;
00701    }
00702    memset(prem_apps_,
00703           mtk::kZero,
00704           sizeof(prem_apps_[0])*num_bndy_coeffs_*dim_null_);
00705
00707
00708    for (auto ll = 0; ll < dim_null_; ++ll) {
00709
00710      // Re-check new generator vector for every iteration except for the first.
00711      #if MTK_DEBUG_LEVEL > 0
00712      if (ll > 0) {
00713        std::cout << "gg" << ll << " =" << std::endl;
00714        for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00715          std::cout << std::setw(12) << gg[ii];
00716        }
00717        std::cout << std::endl << std::endl;
00718      }
00719      #endif
00720
00722      bool transpose{false};
00723
00724
00725      mtk::DenseMatrix AA_(gg,
00726                           num_bndy_coeffs_, order_accuracy_ + 1,
00727                           transpose);
00728
00729      #if MTK_DEBUG_LEVEL > 0
00730      std::cout << "AA_" << ll << " =" << std::endl;
00731      std::cout << AA_ << std::endl;
00732      #endif
00733
00735      mtk::Real *ob{};
00736
00737
```

```
00738      auto ob_ld = num_bndy_coeffs_;
00739
00740      try {
00741        ob = new mtk::Real[ob_ld];
00742      } catch (std::bad_alloc &memory_allocation_exception) {
00743        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00744          std::endl;
00745        std::cerr << memory_allocation_exception.what() << std::endl;
00746      }
00747      memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00748
00749      ob[1] = mtk::kOne;
00750
00751      #if MTK_DEBUG_LEVEL > 0
00752      std::cout << "ob = " << std::endl << std::endl;
00753      for (auto ii = 0; ii < ob_ld; ++ii) {
00754        std::cout << std::setw(12) << ob[ii] << std::endl;
00755      }
00756      std::cout << std::endl;
00757      #endif
00758
00760
00761      // However, this is an under-determined system of equations. So we can not
00762      // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00763      // our LAPACKAdapter class.
00764
00765      int info_{
00766        mtk::LAPACKAdapter::SolveRectangularDenseSystem(AA_,
00767      ob, ob_ld)};
00768      #if MTK_DEBUG_LEVEL > 0
00769      if (!info_) {
00770        std::cout << "System successfully solved!" << std::endl << std::endl;
00771      } else {
00772        std::cerr << "Error solving system! info = " << info_ << std::endl;
00773      }
00774      #endif
00775
00776      #if MTK_DEBUG_LEVEL > 0
00777      std::cout << "ob =" << std::endl;
00778      for (auto ii = 0; ii < ob_ld; ++ii) {
00779        std::cout << std::setw(12) << ob[ii] << std::endl;
00780      }
00781      std::cout << std::endl;
00782      #endif
00783
00785
00786      // This implies a DAXPY operation. However, we must construct the arguments
00787      // for this operation.
00788
00790      // Save them into the ob_bottom array:
00791
00792      Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00793
00794      try {
00795        ob_bottom = new mtk::Real[dim_null_];
00796      } catch (std::bad_alloc &memory_allocation_exception) {
00797        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00798          std::endl;
00799        std::cerr << memory_allocation_exception.what() << std::endl;
00800      }
00801      memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00802
00803      for (auto ii = 0; ii < dim_null_; ++ii) {
00804        ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00805      }
00806
00807      #if MTK_DEBUG_LEVEL > 0
00808      std::cout << "ob_bottom =" << std::endl;
00809      for (auto ii = 0; ii < dim_null_; ++ii) {
00810        std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00811      }
00812      std::cout << std::endl;
00813      #endif
00814
00816
00817      // We must computed an scaled ob, sob, using the scaled null-space in
00818      // rat_basis_null_space_.
00819      // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00820      // or:                ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00821      // thus:              Y =   a*A    *x        +   b*Y (DAXPY).
```

```
00822
00823      #if MTK_DEBUG_LEVEL > 0
00824      std::cout << "Rational basis for the null-space:" << std::endl;
00825      std::cout << rat_basis_null_space_ << std::endl;
00826      #endif
00827
00828      mtk::Real alpha{-mtk::kOne};
00829      mtk::Real beta{mtk::kOne};
00830
00831      mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00832                                   ob_bottom, beta, ob);
00833
00834      #if MTK_DEBUG_LEVEL > 0
00835      std::cout << "scaled ob:" << std::endl;
00836      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00837        std::cout << std::setw(12) << ob[ii] << std::endl;
00838      }
00839      std::cout << std::endl;
00840      #endif
00841
00842      // We save the recently scaled solution, into an array containing these.
00843      // We can NOT start building the pi matrix, simply because I want that part
00844      // to be separated since its construction depends on the algorithm we want
00845      // to implement.
00846
00847      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00848        prem_apps_[ii*dim_null_ + ll] = ob[ii];
00849      }
00850
00851      // After the first iteration, simply shift the entries of the last
00852      // generator vector used:
00853      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00854        gg[ii]--;
00855      }
00856
00857      // Garbage collection for this loop:
00858      delete[] ob;
00859      ob = nullptr;
00860
00861      delete[] ob_bottom;
00862      ob_bottom = nullptr;
00863    } // End of: for (ll = 0; ll < dim_null; ll++);
00864
00865    #if MTK_DEBUG_LEVEL > 0
00866    std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00867    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00868      for (auto jj = 0; jj < dim_null_; ++jj) {
00869        std::cout << std::setw(12) << prem_apps_[ii*dim_null_ + jj];
00870      }
00871      std::cout << std::endl;
00872    }
00873    std::cout << std::endl;
00874    #endif
00875
00876    delete[] gg;
00877    gg = nullptr;
00878
00879    return true;
00880 }
00881
00882 bool mtk::Div1D::ComputeWeights(void) {
00883
00884    // Matrix to copmpute the weights as in the CRSA.
00885    mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00886
00887
00888    // Assemble the pi matrix using:
00889    // 1. The collection of scaled preliminary approximations.
00890    // 2. The collection of coefficients approximating at the interior.
00891    // 3. The scaled basis for the null-space.
00892
00893    // 1.1. Process array of scaled preliminary approximations.
00894
00895    // These are queued in scaled_solutions. Each one of these, will be a column
00896    // of the pi matrix:
00897    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00898      for (auto jj = 0; jj < dim_null_; ++jj) {
00899        pi.data()[ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00900          prem_apps_[ii*dim_null_ + jj];
00901      }
00902    }
00903  }
```

```
00904
00905    // 1.2. Add columns from known stencil approximating at the interior.
00906
00907    // However, these must be padded by zeros, according to their position in the
00908    // final pi matrix:
00909    auto mm = 0;
00910    for (auto jj = dim_null_; jj < order_accuracy_; ++jj) {
00911      for (auto ii = 0; ii < order_accuracy_; ++ii) {
00912        pi.data()[(ii + mm)*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj] =
00913          coeffs_interior_[ii];
00914      }
00915      ++mm;
00916    }
00917
00918    rat_basis_null_space_.OrderColMajor();
00919
00920    #if MTK_DEBUG_LEVEL > 0
00921    std::cout << "Rational basis for the null-space (col. major):" << std::endl;
00922    std::cout << rat_basis_null_space_ << std::endl;
00923    #endif
00924
00925    // 1.3. Add final set of columns: rational basis for null-space.
00926    for (auto jj = dim_null_ + (order_accuracy_/2 + 1); jj < num_bndy_coeffs_ - 1; ++jj) {
00927      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00928        auto og =
00929          (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
00930        auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
00931        pi.data()[de] = rat_basis_null_space_.data()[og];
00932      }
00933    }
00934
00935    #if MTK_DEBUG_LEVEL >0
00936    std::cout << "coeffs_interior_ =" << std::endl;
00937    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00938      std::cout << std::setw(12) << coeffs_interior_[ii];
00939    }
00940    std::cout << std::endl << std::endl;
00941    #endif
00942
00943    #if MTK_DEBUG_LEVEL >0
00944    std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
00945    std::cout << pi << std::endl;
00946    #endif
00947
00948
00949
00950    // This imposes the mimetic condition.
00951
00952    mtk::Real *hh{};  // Right-hand side to compute weights in the C{R,B}SA.
00953
00954    try {
00955      hh = new mtk::Real[num_bndy_coeffs_];
00956    } catch (std::bad_alloc &memory_allocation_exception) {
00957      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00958        std::endl;
00959      std::cerr << memory_allocation_exception.what() << std::endl;
00960    }
00961    memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
00962
00963    hh[0] = -mtk::kOne;
00964    for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
00965      auto aux_xx = mtk::kZero;
00966      for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
00967        aux_xx += coeffs_interior_[jj];
00968      }
00969      hh[ii] = -mtk::kOne*aux_xx;
00970    }
00971
00972
00973
00974    // That is, we construct a system, to solve for the weights.
00975
00976    // Once again we face the challenge of solving with LAPACK. However, for the
00977    // CRSA, this matrix PI is over-determined, since it has more rows than
00978    // unknowns. However, according to the theory, the solution to this system is
00979    // unique. We will use dgels_.
00980
00981    try {
00982      weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
00983    } catch (std::bad_alloc &memory_allocation_exception) {
00984      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00985        std::endl;
00986      std::cerr << memory_allocation_exception.what() << std::endl;
```

```
00987   }
00988   memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
00989
00990   int weights_ld{pi.num_cols() + 1};
00991
00992   // Preserve hh.
00993   std::copy(hh, hh + weights_ld, weights_cbs_);
00994
00995   pi.Transpose();
00996
00997   int info{mtk::LAPACKAdapter::SolveRectangularDenseSystem(
    pi, weights_cbs_, weights_ld)};
00998
00999   #if MTK_DEBUG_LEVEL > 0
01000   if (!info) {
01001     std::cout << "System successfully solved!" << std::endl << std::endl;
01002   } else {
01003     std::cerr << "Error solving system! info = " << info << std::endl;
01004   }
01005   #endif
01006
01007   #if MTK_DEBUG_LEVEL > 0
01008   std::cout << "hh =" << std::endl;
01009   for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01010     std::cout << std::setw(11) << hh[ii] << std::endl;
01011   }
01012   std::cout << std::endl;
01013   #endif
01014
01015   // Preserve the original weights for research.
01016
01017   try {
01018     weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01019   } catch (std::bad_alloc &memory_allocation_exception) {
01020     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01021       std::endl;
01022     std::cerr << memory_allocation_exception.what() << std::endl;
01023   }
01024   memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01025
01026   std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01027
01028   #if MTK_DEBUG_LEVEL > 0
01029   std::cout << "weights_CRSA + lambda =" << std::endl;
01030   for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01031     std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01032   }
01033   std::cout << std::endl;
01034   #endif
01035
01036   if (order_accuracy_ >= mtk::kCriticalOrderAccuracyDiv) {
01037
01039     int minrow_{std::numeric_limits<int>::infinity()};
01040
01041     mtk::Real norm_{mtk::BLASAdapter::RealNRM2(weights_cbs_,
    order_accuracy_)};
01042     mtk::Real minnorm_{std::numeric_limits<mtk::Real>::infinity()};
01043
01045
01046     mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01047
01048     for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01049       for (auto jj = 0; jj < dim_null_; ++jj) {
01050         phi.data()[ii*(order_accuracy_) + jj] = prem_apps_[ii*dim_null_ + jj];
01051       }
01052     }
01053
01054     int aux{};  // Auxiliary variable.
01055     for (auto jj = dim_null_; jj < dim_null_ + 2; ++jj) {
01056       for (auto ii = 0; ii < order_accuracy_; ++ii) {
01057         phi.data()[(ii + aux)*order_accuracy_ + jj] = coeffs_interior_[ii];
01058       }
01059       ++aux;
01060     }
01061
01062     for(auto jj=order_accuracy_ - 1; jj >=order_accuracy_ - dim_null_; jj--) {
01063       for(auto ii=0; ii<order_accuracy_ + 1; ++ii) {
01064         phi.data()[ii*order_accuracy_+jj] = mtk::kZero;
01065       }
01066     }
01067
```

```
01068      for (auto jj = 0; jj < order_accuracy_ + 1; ++jj) {
01069        for (auto ii = 0; ii < dim_null_; ++ii) {
01070          phi.data()[(ii + order_accuracy_ - dim_null_ + jj*order_accuracy_)] =
01071            -prem_apps_[(dim_null_ - ii - 1 + jj*dim_null_)];
01072        }
01073      }
01074
01075      for(auto ii = 0; ii < order_accuracy_/2; ++ii) {
01076        for (auto jj = dim_null_ + 2; jj < order_accuracy_; ++jj) {
01077          auto swap = phi.data()[ii*order_accuracy_+jj];
01078          phi.data()[ii*order_accuracy_ + jj] =
01079            phi.data()[(order_accuracy_-ii)*order_accuracy_+jj];
01080          phi.data()[(order_accuracy_-ii)*order_accuracy_+jj] = swap;
01081        }
01082      }
01083
01084      #if MTK_DEBUG_LEVEL > 0
01085      std::cout << "Constructed PHI matrix for CBS Algorithm: " << std::endl;
01086      std::cout << phi << std::endl;
01087      #endif
01088
01089
01091      mtk::Real *lamed{};  // Used to build big lambda.
01092
01093      try {
01094        lamed = new mtk::Real[dim_null_];
01095      } catch (std::bad_alloc &memory_allocation_exception) {
01096        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01097          std::endl;
01098        std::cerr << memory_allocation_exception.what() << std::endl;
01099      }
01100      memset(lamed, mtk::kZero, sizeof(lamed[0])*dim_null_);
01101
01102      for (auto ii = 0; ii < dim_null_; ++ii) {
01103        lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01104      }
01105
01106      #if MTK_DEBUG_LEVEL > 0
01107      std::cout << "lamed =" << std::endl;
01108      for (auto ii = 0; ii < dim_null_; ++ii) {
01109        std::cout << std::setw(12) << lamed[ii] << std::endl;
01110      }
01111      std::cout << std::endl;
01112      #endif
01113
01114      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01115        mtk::Real temp = mtk::kZero;
01116        for(auto jj = 0; jj < dim_null_; ++jj) {
01117          temp = temp +
01118            lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01119        }
01120        hh[ii] = hh[ii] - temp;
01121      }
01122
01123      #if MTK_DEBUG_LEVEL > 0
01124      std::cout << "big_lambda =" << std::endl;
01125      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01126        std::cout << std::setw(12) << hh[ii] << std::endl;
01127      }
01128      std::cout << std::endl;
01129      #endif
01130
01131      int copy_result{};
01132
01133      mtk::Real normerr_; // Norm of the error for the solution on each row.
01134
01136      for(auto row_= 0; row_ < order_accuracy_ + 1; ++row_) {
01138        normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
          data(),
01139                                                            order_accuracy_ + 1,
01140                                                            order_accuracy_,
01141                                                            order_accuracy_,
01142                                                            hh,
01143                                                            weights_cbs_,
01144                                                            row_,
01145                                                            mimetic_threshold_,
01146                                                            copy_result);
01147        mtk::Real aux{normerr_/norm_};
01148
01149        #if MTK_DEBUG_LEVEL>0
```

```
01150        std::cout << "Relative norm: " << aux << " " << std::endl;
01151        std::cout << std::endl;
01152        #endif
01153
01154        if (aux < minnorm_) {
01155          minnorm_ = aux;
01156          minrow_= row_;
01157        }
01158      }
01159
01160      #if MTK_DEBUG_LEVEL > 0
01161      std::cout << "weights_CBSA + lambda (after brute force search):" <<
01162        std::endl;
01163      for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01164        std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01165      }
01166      std::cout << std::endl;
01167      #endif
01168
01169
01170      // After we know which row yields the smallest relative norm that row is
01171      // chosen to be the objective function and the result of the optimizer is
01172      // chosen to be the new weights_.
01173
01174
01175      #if MTK_DEBUG_LEVEL > 0
01176      std::cout << "Minimum Relative Norm " << minnorm_ << " found at row " <<
01177        minrow_ + 1 << std::endl;
01178      std::cout << std::endl;
01179      #endif
01180
01181      copy_result = 1;
01182      normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
     data(),
01183                                                          order_accuracy_ + 1,
01184                                                          order_accuracy_,
01185                                                          order_accuracy_,
01186                                                          hh,
01187                                                          weights_cbs_,
01188                                                          minrow_,
01189                                                          mimetic_threshold_,
01190                                                          copy_result);
01191      mtk::Real aux_{normerr_/norm_};
01192      #if MTK_DEBUG_LEVEL > 0
01193      std::cout << "Relative norm: " << aux_ << std::endl;
01194      std::cout << std::endl;
01195      #endif
01196
01197      delete [] lamed;
01198      lamed = nullptr;
01199    }
01200
01201    delete [] hh;
01202    hh = nullptr;
01203
01204    return true;
01205 }
01206
01207 bool mtk::Div1D::ComputeStencilBoundaryGrid(void) {
01208
01209    #if MTK_DEBUG_LEVEL > 0
01210    std::cout << "weights_CBSA + lambda =" << std::endl;
01211    for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01212      std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01213    }
01214    std::cout << std::endl;
01215    #endif
01216
01218
01219    mtk::Real *lambda{}; // Collection of bottom values from weights_.
01220
01221    try {
01222      lambda = new mtk::Real[dim_null_];
01223    } catch (std::bad_alloc &memory_allocation_exception) {
01224      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01225        std::endl;
01226      std::cerr << memory_allocation_exception.what() << std::endl;
01227    }
01228    memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01229
01230    for (auto ii = 0; ii < dim_null_; ++ii) {
01231      lambda[ii] = weights_cbs_[order_accuracy_ + ii];
```

```
01232   }
01233
01234   #if MTK_DEBUG_LEVEL > 0
01235   std::cout << "lambda =" << std::endl;
01236   for (auto ii = 0; ii < dim_null_; ++ii) {
01237     std::cout << std::setw(12) << lambda[ii] << std::endl;
01238   }
01239   std::cout << std::endl;
01240   #endif
01241
01242
01243
01244   mtk::Real *alpha{}; // Collection of alpha values.
01245
01246   try {
01247     alpha = new mtk::Real[dim_null_];
01248   } catch (std::bad_alloc &memory_allocation_exception) {
01249     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01250       std::endl;
01251     std::cerr << memory_allocation_exception.what() << std::endl;
01252   }
01253   memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01254
01255   for (auto ii = 0; ii < dim_null_; ++ii) {
01256     alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01257   }
01258
01259   #if MTK_DEBUG_LEVEL > 0
01260   std::cout << "alpha =" << std::endl;
01261   for (auto ii = 0; ii < dim_null_; ++ii) {
01262     std::cout << std::setw(12) << alpha[ii] << std::endl;
01263   }
01264   std::cout << std::endl;
01265   #endif
01266
01267
01268
01269   try {
01270     mim_bndy_ = new mtk::Real[num_bndy_coeffs_*dim_null_];
01271   } catch (std::bad_alloc &memory_allocation_exception) {
01272     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01273       std::endl;
01274     std::cerr << memory_allocation_exception.what() << std::endl;
01275   }
01276   memset(mim_bndy_, mtk::kZero, sizeof(mim_bndy_[0])*num_bndy_coeffs_*dim_null_);
01277
01278   for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01279     for (auto jj = 0; jj < dim_null_; ++jj) {
01280       mim_bndy_[ii*dim_null_ + jj] =
01281         prem_apps_[ii*dim_null_ + jj] +
01282         alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01283     }
01284   }
01285
01286   #if MTK_DEBUG_LEVEL >0
01287   std::cout << "Collection of mimetic approximations:" << std::endl;
01288   for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01289     for (auto jj = 0; jj < dim_null_; ++jj) {
01290       std::cout << std::setw(13) << mim_bndy_[ii*dim_null_ + jj];
01291     }
01292     std::cout << std::endl;
01293   }
01294   std::cout << std::endl;
01295   #endif
01296
01297   delete[] lambda;
01298   lambda = nullptr;
01299
01300   delete[] alpha;
01301   alpha = nullptr;
01302
01303   return true;
01304 }
01305
01306 bool mtk::Div1D::AssembleOperator(void) {
01307
01308   // The output array will have this form:
01309   // 1. The first entry of the array will contain the used order order_accuracy_.
01310   // 2. The second entry of the array will contain the collection of
01311   // approximating coefficients for the interior of the grid.
01312   // 3. IF order_accuracy_ > 2, then the third entry will contain a collection of weights.
01313   // 4. IF order_accuracy_ > 2, the next dim_null_ entries will contain the collections of
01314   // approximating coefficients for the west boundary of the grid.
```

```
01315
01316    if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01317      divergence_length_ =
01318        1 + order_accuracy_ + order_accuracy_ + dim_null_*num_bndy_coeffs_;
01319    } else {
01320      divergence_length_ = 1 + order_accuracy_;
01321    }
01322
01323    #if MTK_DEBUG_LEVEL > 0
01324    std::cout << "divergence_length_ = " << divergence_length_ << std::endl;
01325    #endif
01326
01327    try {
01328      divergence_ = new double[divergence_length_];
01329    } catch (std::bad_alloc &memory_allocation_exception) {
01330      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01331        std::endl;
01332      std::cerr << memory_allocation_exception.what() << std::endl;
01333    }
01334    memset(divergence_, mtk::kZero, sizeof(divergence_[0])*divergence_length_);
01335
01337
01338    divergence_[0] = order_accuracy_;
01339
01341
01342    for (auto ii = 0; ii < order_accuracy_; ++ii) {
01343    divergence_[ii + 1] = coeffs_interior_[ii];
01344    }
01345
01347
01348    if (order_accuracy_ > 2) {
01349      for (auto ii = 0; ii < order_accuracy_; ++ii) {
01350        divergence_[(1 + order_accuracy_) + ii] = weights_cbs_[ii];
01351      }
01352    }
01353
01356
01357    if (order_accuracy_ > 2) {
01358      auto offset = (2*order_accuracy_ + 1);
01359      int mm{};
01360      for (auto ii = 0; ii < dim_null_; ++ii) {
01361        for (auto jj = 0; jj < num_bndy_coeffs_; ++jj) {
01362          divergence_[offset + (mm)] = mim_bndy_[jj*dim_null_ + ii];
01363          ++mm;
01364        }
01365      }
01366    }
01367
01368    #if MTK_DEBUG_LEVEL > 0
01369    std::cout << "1D " << order_accuracy_ << "-order div built!" << std::endl;
01370    std::cout << std::endl;
01371    #endif
01372
01373    return true;
01374 }
```

17.41 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:
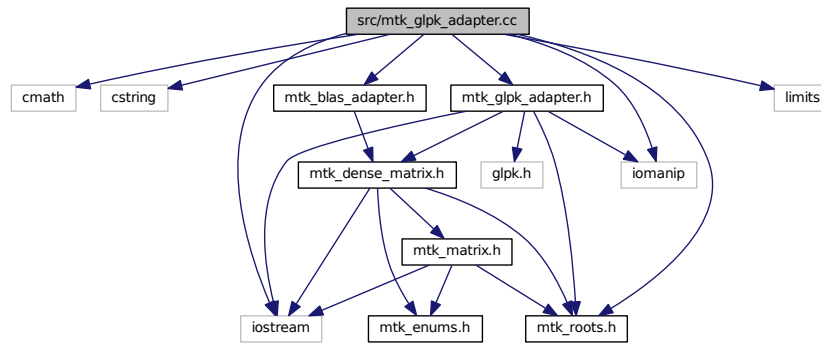
17.41.1 Detailed Description

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

Todo Document better this file.

Definition in file mtk_glpk_adapter.cc.

17.42 mtk_glpk_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. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
```

```
00036 list of conditions and the following disclaimer.
00037
00038 4. Redistributions in binary form must reproduce the above copyright notice,
00039 this list of conditions and the following disclaimer in the documentation and/or
00040 other materials provided with the distribution.
00041
00042 5. Usage of the binary form on proprietary applications shall require explicit
00043 prior written permission from the the copyright holders.
00044
00045 6. 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 <cmath>
00068 #include <cstring>
00069
00070 #include <iostream>
00071 #include <iomanip>
00072 #include <limits>
00073
00074 #include "mtk_roots.h"
00075 #include "mtk_blas_adapter.h"
00076 #include "mtk_glpk_adapter.h"
00077
00078 mtk::Real mtk::GLPKAdapter::SolveSimplexAndCompare(
       mtk::Real *A,
00079                                                   int nrows,
00080                                                   int ncols,
00081                                                   int kk,
00082                                                   mtk::Real *hh,
00083                                                   mtk::Real *qq,
00084                                                   int robjective,
00085                                                   mtk::Real mimetic_threshold,
00086                                                   int copy) {
00087
00088   #if MTK_DEBUG_LEVEL > 0
00089   char mps_file_name[18]; //  File name for the MPS files.
00090   #endif
00091   char rname[5];          //
00092   char cname[5];          //
00093
00094   glp_prob *lp; // Linear programming problem.
00095
00096   int *ia;  //
00097   int *ja;  //
00098
00099   int problem_size; // Size of the problem.
00100   int lp_nrows;     // Number of rows.
00101   int lp_ncols;     // Number of columns.
00102   int matsize;      //
00103   int glp_index{1};    // Index of the objective function.
00104   int ii;           //
00105   int jj;           //
00106
00107   mtk::Real *ar;          //
00108   mtk::Real *objective;   //
00109   mtk::Real *rhs;         //
00110   mtk::Real *err;         //
00111   mtk::Real x1;           //
00112
00113   #if MTK_DEBUG_LEVEL > 0
00114   mtk::Real obj_value;    //
00115   #endif
```

```
00116
00117   lp_nrows = kk;
00118   lp_ncols = kk;
00119
00120   matsize = lp_nrows*lp_ncols;
00121
00123
00125   problem_size = lp_nrows*lp_ncols + 1;
00126
00127   try {
00128     ia = new int[problem_size];
00129   } catch (std::bad_alloc &memory_allocation_exception) {
00130     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00131       std::endl;
00132     std::cerr << memory_allocation_exception.what() << std::endl;
00133   }
00134   memset(ia, 0, sizeof(ia[0])*problem_size);
00135
00136   try {
00137     ja = new int[problem_size];
00138   } catch (std::bad_alloc &memory_allocation_exception) {
00139     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00140       std::endl;
00141     std::cerr << memory_allocation_exception.what() << std::endl;
00142   }
00143   memset(ja, 0, sizeof(ja[0])*problem_size);
00144
00145   try {
00146     ar = new mtk::Real[problem_size];
00147   } catch (std::bad_alloc &memory_allocation_exception) {
00148     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00149       std::endl;
00150     std::cerr << memory_allocation_exception.what() << std::endl;
00151   }
00152   memset(ar, mtk::kZero, sizeof(ar[0])*problem_size);
00153
00154   try {
00155     objective = new mtk::Real[lp_ncols + 1];
00156   } catch (std::bad_alloc &memory_allocation_exception) {
00157     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00158       std::endl;
00159     std::cerr << memory_allocation_exception.what() << std::endl;
00160   }
00161   memset(objective, mtk::kZero, sizeof(objective[0])*(lp_ncols + 1));
00162
00163   try {
00164     rhs = new mtk::Real[lp_nrows + 1];
00165   } catch (std::bad_alloc &memory_allocation_exception) {
00166     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00167       std::endl;
00168     std::cerr << memory_allocation_exception.what() << std::endl;
00169   }
00170   memset(rhs, mtk::kZero, sizeof(rhs[0])*(lp_nrows + 1));
00171
00172   try {
00173     err = new mtk::Real[lp_nrows];
00174   } catch (std::bad_alloc &memory_allocation_exception) {
00175     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00176       std::endl;
00177     std::cerr << memory_allocation_exception.what() << std::endl;
00178   }
00179   memset(err, mtk::kZero, sizeof(err[0])*(lp_nrows));
00180
00181   #if MTK_DEBUG_LEVEL > 0
00182   std::cout << "Problem size: " << problem_size << std::endl;
00183   std::cout << "lp_nrows = " << lp_nrows << std::endl;
00184   std::cout << "lp_ncols = " << lp_ncols << std::endl;
00185   std::cout << std::endl;
00186   #endif
00187
00188   lp = glp_create_prob();
00189
00190   glp_set_prob_name (lp, "mtk::GLPKAdapter::Simplex");
00191
00192   glp_set_obj_dir (lp, GLP_MIN);
00193
00195
00196   glp_add_rows(lp, lp_nrows);
00197
00198   for (ii = 1; ii <= lp_nrows; ++ii) {
00199     sprintf(rname, "R%02d",ii);
```

```
00200      glp_set_row_name(lp, ii, rname);
00201   }
00202
00203   glp_add_cols(lp, lp_ncols);
00204
00205   for (ii = 1; ii <= lp_ncols; ++ii) {
00206     sprintf(cname, "Q%02d",ii);
00207     glp_set_col_name (lp, ii, cname);
00208   }
00209
00210
00211
00212   #if MTK_DEBUG_LEVEL>0
00213   std::cout << "Using row " << robjective + 1 << " as objective." << std::endl;
00214   #endif
00215   for (jj = 0; jj < kk; ++jj) {
00216     objective[glp_index] = A[jj + robjective * ncols];
00217     glp_index++;
00218   }
00219   #if MTK_DEBUG_LEVEL >0
00220   std::cout << std::endl;
00221   #endif
00222
00224
00225   glp_index = 1;
00226   rhs[0] = mtk::kZero;
00227   for (ii = 0; ii <= lp_nrows; ++ii) {
00228     if (ii != robjective) {
00229       rhs[glp_index] = hh[ii];
00230       glp_set_row_bnds(lp, glp_index, GLP_UP, 0.0, rhs[glp_index]);
00231       glp_index++;
00232     }
00233   }
00234
00235   #if MTK_DEBUG_LEVEL > 0
00236   std::cout << "rhs =" << std::endl;
00237   for (auto ii = 0; ii < lp_nrows; ++ii) {
00238     std::cout << std::setw(15) << rhs[ii] << std::endl;
00239   }
00240   std::cout << std::endl;
00241   #endif
00242
00244   for (ii = 1; ii <= lp_ncols; ++ii) {
00245     glp_set_obj_coef (lp, ii, objective[ii]);
00246   }
00247   }
00248
00250
00251   for (ii = 1; ii <= lp_ncols; ++ii) {
00252     glp_set_col_bnds (lp, ii, GLP_LO, mimetic_threshold, 0.0);
00253   }
00254
00256
00257   glp_index = 1;
00258   for (ii = 0; ii <= kk; ++ii) {
00259     for (jj = 0; jj < kk; ++jj) {
00260       if (ii != robjective) {
00261         ar[glp_index] = A[jj + ii * ncols];
00262         glp_index++;
00263       }
00264     }
00265   }
00266
00267   glp_index = 0;
00268
00269   for (ii = 1; ii < problem_size; ++ii) {
00270     if (((ii - 1) % lp_ncols) == 0) {
00271       glp_index++;
00272     }
00273     ia[ii] = glp_index;
00274     ja[ii] = (ii - 1) % lp_ncols + 1;
00275   }
00276
00277   glp_load_matrix (lp, matsize, ia, ja, ar);
00278
00279   #if MTK_DEBUG_LEVEL > 0
00280   sprintf(mps_file_name, "LP_MPS_row_%02d.mps", robjective);
00281   glp_write_mps(lp, GLP_MPS_FILE, nullptr, mps_file_name);
00282   #endif
00283
00285
00286   glp_simplex (lp, nullptr);
```

```
00287
00288    // Check status of the solution.
00289
00290    if (glp_get_status(lp) == GLP_OPT) {
00291
00292      for(ii = 1; ii <= lp_ncols; ++ii) {
00293        err[ii - 1] = qq[ii - 1] - glp_get_col_prim(lp,ii);
00294      }
00295
00296      #if MTK_DEBUG_LEVEL > 0
00297      obj_value = glp_get_obj_val (lp);
00298      std::cout << std::setw(12) << "CBS" << std::setw(12) << "CRS" << std::endl;
00299      for (ii = 0; ii < lp_ncols; ++ii) {
00300        std::cout << "q_" << ii + 1 << " = " << std::setw(12) <<
00301          glp_get_col_prim(lp,ii + 1) << std::setw(12) << qq[ii] << std::endl;
00302      }
00303      std::cout << "Objective function value (row " << robjective + 1 << ") = " <<
00304        obj_value << std::endl;
00305      #endif
00306
00307      if (copy) {
00308        for(ii = 0; ii < lp_ncols; ++ii) {
00309          qq[ii] = glp_get_col_prim(lp,ii + 1);
00310        }
00311        // Preserve the bottom values of qq.
00312      }
00313
00314      x1 = mtk::BLASAdapter::RealNRM2(err,lp_ncols);
00315
00316    } else {
00317      x1 = std::numeric_limits<mtk::Real>::infinity();
00318    }
00319
00320    glp_delete_prob (lp);
00321    glp_free_env ();
00322
00323    delete [] ia;
00324    delete [] ja;
00325    delete [] ar;
00326    delete [] objective;
00327    delete [] rhs;
00328    delete [] err;
00329
00330    return x1;
00331 }
```

17.43 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:
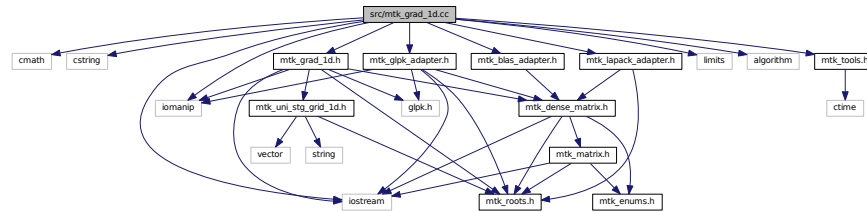
Namespaces

- **mtk**

  Mimetic Methods Toolkit namespace.

Functions

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

17.43.1   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.

17.44   mtk_grad_1d.cc

```
00001
00015 /*
00016 Copyright (C) 2015, Computational Science Research Center, San Diego State
00017 University. All rights reserved.
00018
00019 Redistribution and use in source and binary forms, with or without modification,
00020 are permitted provided that the following conditions are met:
00021
00022 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00023 and a copy of the modified files should be reported once modifications are
00024 completed. Documentation related to said modifications should be included.
00025
00026 2. Redistributions of source code must be done through direct
00027 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
```

```
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
00038
00039 6. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cmath>
00062 #include <cstring>
00063
00064 #include <iostream>
00065 #include <iomanip>
00066 #include <limits>
00067 #include <algorithm>
00068
00069 #include "mtk_tools.h"
00070
00071 #include "mtk_blas_adapter.h"
00072 #include "mtk_lapack_adapter.h"
00073 #include "mtk_glpk_adapter.h"
00074
00075 #include "mtk_grad_1d.h"
00076
00077 namespace mtk {
00078
00079 std::ostream& operator <<(std::ostream &stream, mtk::Grad1D &in) {
00080
00082   stream << "gradient_[0] = " << std::setw(9) << in.gradient_[0] << std::endl;
00083
00084
00086   stream << "gradient_[1:" << in.order_accuracy_ << "] = ";
00087   for (auto ii = 1; ii <= in.order_accuracy_; ++ii) {
00088     stream << std::setw(9) << in.gradient_[ii] << " ";
00089   }
00090   stream << std::endl;
00091
00092
00094   stream << "gradient_[" << in.order_accuracy_ + 1 << ":" <<
00095     2*in.order_accuracy_ << "] = ";
00096   for (auto ii = in.order_accuracy_ + 1; ii <= 2*in.
00097   order_accuracy_; ++ii) {
00098     stream << std::setw(9) << in.gradient_[ii] << " ";
00099   }
00100   stream << std::endl;
00101
00103
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    #if MTK_DEBUG_LEVEL > 0
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
00205    #if MTK_DEBUG_LEVEL > 0
00206    if (!abort_construction) {
00207      std::cerr << "Could NOT complete stage 1." << std::endl;;
00208      std::cerr << "Exiting..." << std::endl;;
00209      return false;
00210    }
00211    #endif
00212
00213    // At this point, we already have the values for the interior stencil stored
00214    // in the coeffs_interior_ array.
00215
00216    dim_null_ = order_accuracy_/2 - 1;
00217
00218    num_bndy_approxs_ = dim_null_ + 1;
00219
00220    #ifdef MTK_PRECISION_DOUBLE
00221    num_bndy_coeffs_ = (int) (3.0*((mtk::Real) order_accuracy_)/2.0);
00222    #else
00223    num_bndy_coeffs_ = (int) (3.0f*((mtk::Real) order_accuracy_)/2.0f);
00224    #endif
00225
00227
00228    // For this we will follow recommendations given in:
00229    //
00230    // http://icl.cs.utk.edu/lapack-forum/viewtopic.php?f=5&t=4506
00231    //
00232    // We will compute the QR Factorization of the transpose, as in the
00233    // following (MATLAB) pseudo-code:
00234    //
00235    // [Q,R] = qr(V'); % Full QR as defined in
00236    // % http://www.stanford.edu/class/ee263/notes/qr_matlab.pdf
00237    //
00238    // null-space = Q(:, last (order_accuracy_/2 - 1) columns of Q );
00239    //
00240    // However, given the nature of the Vandermonde matrices we've just
00241    // computed, they all posses the same null-space. Therefore, we impose the
00242    // convention of computing the null-space of the first Vandermonde matrix
00243    // (west boundary).
00244
00245    // In the case of the gradient, the first Vandermonde system has a unique
00246    // solution for the case of second-order-accuracy. Ergo, the Vandermonde
00247    // matrix used to assemble said system, will have an empty null-space.
00248
00249    // Therefore, we only compute a rational basis for the case of order higher
00250    // than second.
00251
00252    if (dim_null_ > 0) {
00253
00254      abort_construction = ComputeRationalBasisNullSpace();
00255
00256      #if MTK_DEBUG_LEVEL > 0
00257      if (!abort_construction) {
00258        std::cerr << "Could NOT complete stage 2.1." << std::endl;;
00259        std::cerr << "Exiting..." << std::endl;;
00260        return false;
00261      }
00262      #endif
00263    }
00264
00266
00267    abort_construction = ComputePreliminaryApproximations();
00268
00269    #if MTK_DEBUG_LEVEL > 0
00270    if (!abort_construction) {
00271      std::cerr << "Could NOT complete stage 2.2." << std::endl;;
00272      std::cerr << "Exiting..." << std::endl;;
00273      return false;
00274    }
00275    #endif
00276
00278
00279    abort_construction = ComputeWeights();
```

```
00280
00281    #if MTK_DEBUG_LEVEL > 0
00282    if (!abort_construction) {
00283      std::cerr << "Could NOT complete stage 2.3." << std::endl;;
00284      std::cerr << "Exiting..." << std::endl;;
00285      return false;
00286    }
00287    #endif
00288
00290
00291    if (dim_null_ > 0) {
00292
00293      abort_construction = ComputeStencilBoundaryGrid();
00294
00295      #if MTK_DEBUG_LEVEL > 0
00296      if (!abort_construction) {
00297        std::cerr << "Could NOT complete stage 2.4." << std::endl;;
00298        std::cerr << "Exiting..." << std::endl;;
00299        return false;
00300      }
00301      #endif
00302    }
00303
00305
00306    // Once we have the following three collections of data:
00307    //   (a) the coefficients for the interior,
00308    //   (b) the coefficients for the boundary (if it applies),
00309    //   (c) and the weights (if it applies),
00310    // we will store everything in the output array:
00311
00312    abort_construction = AssembleOperator();
00313
00314    #if MTK_DEBUG_LEVEL > 0
00315    if (!abort_construction) {
00316      std::cerr << "Could NOT complete stage 3." << std::endl;;
00317      std::cerr << "Exiting..." << std::endl;;
00318      return false;
00319    }
00320    #endif
00321
00322    return true;
00323  }
00324
00325  mtk::Real *mtk::Grad1D::weights_crs() {
00326
00327    return weights_crs_;
00328  }
00329
00330  mtk::Real *mtk::Grad1D::weights_cbs() {
00331
00332    return weights_cbs_;
00333  }
00334
00335  mtk::DenseMatrix mtk::Grad1D::ReturnAsDenseMatrix(const
      UniStgGrid1D &grid) {
00336
00337    int nn{grid.num_cells_x()}; // Number of cells on the grid.
00338
00339    #if MTK_DEBUG_LEVEL > 0
00340    mtk::Tools::Prevent(order_accuracy_ <= 0, __FILE__, __LINE__, __func__);
00341
00343
00344    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 2, __FILE__, __LINE__, __func__);
00345    #endif
00346
00347    mtk::Real inv_delta_x{mtk::kOne/grid.delta_x()};
00348
00349    int gg_num_rows = nn + 1;
00350    int gg_num_cols = nn + 2;
00351    int elements_per_row = num_bndy_coeffs_;
00352    int num_extra_rows = order_accuracy_/2;
00353
00354    // Output matrix featuring sizes for gradient operators.
00355    mtk::DenseMatrix out(gg_num_rows, gg_num_cols);
00356
00358
00359    auto ee_index = 0;
00360    for (auto ii = 0; ii < num_extra_rows; ii++) {
00361      auto cc = 0;
00362      for(auto jj = 0 ; jj < gg_num_cols; jj++) {
00363        if(cc >= elements_per_row) {
```

```
00364          out.SetValue(ii, jj, mtk::kZero);
00365        } else {
00366          out.SetValue(ii,jj,
00367                       gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00368          cc++;
00369        }
00370      }
00371    }
00372
00374
00375    for (auto ii = num_extra_rows; ii < gg_num_rows - num_extra_rows; ii++) {
00376      auto jj = ii - num_extra_rows + 1;
00377      for (auto cc = 0; cc < order_accuracy_; cc++, jj++) {
00378        out.SetValue(ii, jj, coeffs_interior_[cc]*inv_delta_x);
00379      }
00380    }
00381
00383
00384    ee_index = 0;
00385    for (auto ii = gg_num_rows - 1; ii >= gg_num_rows - num_extra_rows; ii--) {
00386      auto cc = 0;
00387      for (auto jj = gg_num_cols - 1; jj >= 0; jj--) {
00388        if(cc >= elements_per_row) {
00389          out.SetValue(ii,jj,mtk::kZero);
00390        } else {
00391          out.SetValue(ii,jj,
00392                       -gradient_[2*order_accuracy_ + 1 + ee_index++]*inv_delta_x);
00393          cc++;
00394        }
00395      }
00396    }
00397
00398    return out;
00399 }
00400
00401 bool mtk::Grad1D::ComputeStencilInteriorGrid() {
00402
00404
00405    mtk::Real* pp{}; // Spatial coordinates to create interior stencil.
00406
00407    try {
00408      pp = new mtk::Real[order_accuracy_];
00409    } catch (std::bad_alloc &memory_allocation_exception) {
00410      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00411        std::endl;
00412      std::cerr << memory_allocation_exception.what() << std::endl;
00413    }
00414    memset(pp, mtk::kZero, sizeof(pp[0])*order_accuracy_);
00415
00416    #ifdef MTK_PRECISION_DOUBLE
00417    pp[0] = 1.0/2.0 - ((mtk::Real) order_accuracy_)/2.0;
00418    #else
00419    pp[0] = 1.0f/2.0f - ((mtk::Real) order_accuracy_)/2.0f;
00420    #endif
00421
00422    for (auto ii = 1; ii < order_accuracy_; ++ii) {
00423      pp[ii] = pp[ii - 1] + mtk::kOne;
00424    }
00425
00426    #if MTK_DEBUG_LEVEL > 0
00427    std::cout << "pp =" << std::endl;
00428    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00429      std::cout << std::setw(12) << pp[ii];
00430    }
00431    std::cout << std::endl << std::endl;
00432    #endif
00433
00435
00436    bool transpose{false};
00437
00438    mtk::DenseMatrix vander_matrix(pp,order_accuracy_,order_accuracy_,transpose);
00439
00440    #if MTK_DEBUG_LEVEL > 0
00441    std::cout << "vander_matrix = " << std::endl;
00442    std::cout << vander_matrix << std::endl << std::endl;
00443    #endif
00444
00446
00447    try {
00448      coeffs_interior_ = new mtk::Real[order_accuracy_];
00449    } catch (std::bad_alloc &memory_allocation_exception) {
```

```
00450    std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00451       std::endl;
00452    std::cerr << memory_allocation_exception.what() << std::endl;
00453    }
00454    memset(coeffs_interior_, mtk::kZero, sizeof(coeffs_interior_[0])*order_accuracy_);
00455
00456    coeffs_interior_[1] = mtk::kOne;
00457
00458    #if MTK_DEBUG_LEVEL > 0
00459    std::cout << "oo =" << std::endl;
00460    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00461      std::cout << std::setw(12) << coeffs_interior_[ii] << std::endl;
00462    }
00463    std::cout << std::endl;
00464    #endif
00465
00466
00467
00468    int info{mtk::LAPACKAdapter::SolveDenseSystem(vander_matrix,
00469                                                 coeffs_interior_)};
00470
00471    #if MTK_DEBUG_LEVEL > 0
00472    if (!info) {
00473      std::cout << "System solved! Interior stencil attained!" << std::endl;
00474      std::cout << std::endl;
00475    }
00476    else {
00477      std::cerr << "Something wrong solving system! info = " << info << std::endl;
00478      std::cerr << "Exiting..." << std::endl;
00479      return false;
00480    }
00481    #endif
00482
00483    #if MTK_DEBUG_LEVEL > 0
00484    std::cout << "coeffs_interior_ =" << std::endl;
00485    for (auto ii = 0; ii < order_accuracy_; ++ii) {
00486      std::cout << std::setw(12) << coeffs_interior_[ii];
00487    }
00488    std::cout << std::endl << std::endl;
00489    #endif
00490
00491    delete [] pp;
00492    pp = nullptr;
00493
00494    return true;
00495 }
00496
00497 bool mtk::Grad1D::ComputeRationalBasisNullSpace(void) {
00498
00499
00500
00501    mtk::Real* gg{}; // Generator vector for the first Vandermonde matrix.
00502
00503    try {
00504      gg = new mtk::Real[num_bndy_coeffs_];
00505    } catch (std::bad_alloc &memory_allocation_exception) {
00506      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00507        std::endl;
00508      std::cerr << memory_allocation_exception.what() << std::endl;
00509    }
00510    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00511
00512    #ifdef MTK_PRECISION_DOUBLE
00513    gg[1] = 1.0/2.0;
00514    #else
00515    gg[1] = 1.0f/2.0f;
00516    #endif
00517    for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00518      gg[ii] = gg[ii - 1] + mtk::kOne;
00519    }
00520
00521    #if MTK_DEBUG_LEVEL > 0
00522    std::cout << "gg =" << std::endl;
00523    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00524      std::cout << std::setw(12) << gg[ii];
00525    }
00526    std::cout << std::endl << std::endl;
00527    #endif
00528
00529
00530
00531    bool tran{true}; // Should I transpose the Vandermonde matrix.
00532
00533    mtk::DenseMatrix aa_west_t(gg, num_bndy_coeffs_, order_accuracy_ + 1, tran);
```

```
00534
00535    #if MTK_DEBUG_LEVEL > 0
00536    std::cout << "aa_west_t =" << std::endl;
00537    std::cout << aa_west_t << std::endl;
00538    #endif
00539
00540
00541
00542    mtk::DenseMatrix qq_t(mtk::LAPACKAdapter::QRFactorDenseMatrix
       (aa_west_t));
00543
00544    #if MTK_DEBUG_LEVEL > 0
00545    std::cout << "qq_t = " << std::endl;
00546    std::cout << qq_t << std::endl;
00547    #endif
00548
00549
00550
00551    int kk_num_rows{num_bndy_coeffs_};
00552    int kk_num_cols{dim_null_};
00553
00554    mtk::DenseMatrix kk(kk_num_rows, kk_num_cols);
00555
00556    // In the case of the gradient, even though we must solve for a null-space
00557    // of dimension 2, we must only extract ONE basis for the kernel.
00558    // We perform this extraction here:
00559
00560    int aux_{kk_num_rows - kk_num_cols};
00561    for (auto ii = kk_num_rows - kk_num_cols; ii < kk_num_rows; ii++) {
00562      aux_--;
00563      for (auto jj = 0; jj < kk_num_rows; jj++) {
00564        kk.data()[jj*kk_num_cols + (kk_num_rows - kk_num_cols - aux_ - 1)] =
00565          qq_t.data()[ii*num_bndy_coeffs_ + jj];
00566      }
00567    }
00568
00569    #if MTK_DEBUG_LEVEL > 0
00570    std::cout << "kk =" << std::endl;
00571    std::cout << kk << std::endl;
00572    std::cout << "kk.num_rows() = " << kk.num_rows() << std::endl;
00573    std::cout << "kk.num_cols() = " << kk.num_cols() << std::endl;
00574    std::cout << std::endl;
00575    #endif
00576
00577
00578
00579    // Scale thus requesting that the last entries of the attained basis for the
00580    // null-space, adopt the pattern we require.
00581    // Essentially we will implement the following MATLAB pseudo-code:
00582    //   scalers = kk(num_bndy_approxs - (dim_null - 1):num_bndy_approxs,:)\B
00583    //   SK = kk*scalers
00584    // where SK is the scaled null-space.
00585
00586    // In this point, we almost have all the data we need correctly allocated
00587    // in memory. We will create the matrix iden_, and elements we wish to scale in
00588    // the kk array. Using the concept of the leading dimension, we could just
00589    // use kk, with the correct leading dimension and that is it. BUT I DO NOT
00590    // GET how does it work. So I will just create a matrix with the content of
00591    // this array that we need, solve for the scalers and then scale the
00592    // whole kk:
00593
00594    // We will then create memory for that sub-matrix of kk (subk).
00595
00596    mtk::DenseMatrix subk(dim_null_, dim_null_);
00597
00598    auto zz = 0;
00599    for (auto ii = order_accuracy_ + 1; ii < num_bndy_coeffs_; ii++) {
00600      for (auto jj = 0; jj < dim_null_; jj++) {
00601        subk.data()[zz*(dim_null_) + jj] = kk.data()[ii*(dim_null_) + jj];
00602      }
00603      zz++;
00604    }
00605
00606    #if MTK_DEBUG_LEVEL > 0
00607    std::cout << "subk =" << std::endl;
00608    std::cout << subk << std::endl;
00609    #endif
00610
00611    subk.Transpose();
00612
00613    #if MTK_DEBUG_LEVEL > 0
00614    std::cout << "subk_t =" << std::endl;
00615    std::cout << subk << std::endl;
00616    #endif
```

```
00617
00618    bool padded{false};
00619    tran = false;
00620
00621    mtk::DenseMatrix iden(dim_null_, padded, tran);
00622
00623    #if MTK_DEBUG_LEVEL > 0
00624    std::cout << "iden =" << std::endl;
00625    std::cout << iden << std::endl;
00626    #endif
00627
00628    // Solve the system to compute the scalers.
00629    // An example of the system to solve, for k = 8, is:
00630    //
00631    // subk*scalers = iden or
00632    //
00633    // |  0.386018  -0.0339244   -0.129478 |            | 1 0 0 |
00634    // | -0.119774   0.0199423   0.0558632 |*scalers = | 0 1 0 |
00635    // | 0.0155708 -0.00349546 -0.00853182 |            | 0 0 1 |
00636    //
00637    // Notice this is a nrhs = 3 system.
00638    // Noteworthy: we do NOT ACTUALLY ALLOCATE space for the scalers... they
00639    // will be stored in the created identity matrix.
00640    // Let us first transpose subk (because of LAPACK):
00641
00642    int info{mtk::LAPACKAdapter::SolveDenseSystem(subk, iden)};
00643
00644    #if MTK_DEBUG_LEVEL > 0
00645    if (!info) {
00646      std::cout << "System successfully solved!" <<
00647        std::endl;
00648    } else {
00649      std::cerr << "Something went wrong solving system! info = " << info <<
00650        std::endl;
00651      std::cerr << "Exiting..." << std::endl;
00652      return false;
00653    }
00654    std::cout << std::endl;
00655    #endif
00656
00657    #if MTK_DEBUG_LEVEL > 0
00658    std::cout << "Computed scalers:" << std::endl;
00659    std::cout << iden << std::endl;
00660    #endif
00661
00662    // Multiply the two matrices to attain a scaled basis for null-space.
00663
00664    rat_basis_null_space_ = mtk::BLASAdapter::RealDenseMM(kk, iden);
00665
00666    #if MTK_DEBUG_LEVEL > 0
00667    std::cout << "Rational basis for the null-space:" << std::endl;
00668    std::cout << rat_basis_null_space_ << std::endl;
00669    #endif
00670
00671    // At this point, we have a rational basis for the null-space, with the
00672    // pattern we need! :)
00673
00674    delete [] gg;
00675    gg = nullptr;
00676
00677    return true;
00678 }
00679
00680 bool mtk::Grad1D::ComputePreliminaryApproximations() {
00681
00682
00683    mtk::Real *gg{}; // Generator vector for the first approximation.
00684
00685
00686    try {
00687      gg = new mtk::Real[num_bndy_coeffs_];
00688    } catch (std::bad_alloc &memory_allocation_exception) {
00689      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00690        std::endl;
00691      std::cerr << memory_allocation_exception.what() << std::endl;
00692    }
00693    memset(gg, mtk::kZero, sizeof(gg[0])*num_bndy_coeffs_);
00694
00695    #ifdef MTK_PRECISION_DOUBLE
00696    gg[1] = 1.0/2.0;
00697    #else
00698    gg[1] = 1.0f/2.0f;
```

```
00699    #endif
00700    for (auto ii = 2; ii < num_bndy_coeffs_; ++ii) {
00701      gg[ii] = gg[ii - 1] + mtk::kOne;
00702    }
00703
00704    #if MTK_DEBUG_LEVEL > 0
00705    std::cout << "gg0 =" << std::endl;
00706    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00707      std::cout << std::setw(12) << gg[ii];
00708    }
00709    std::cout << std::endl << std::endl;
00710    #endif
00711
00712    // Allocate 2D array to store the collection of preliminary approximations.
00713    try {
00714      prem_apps_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
00715    } catch (std::bad_alloc &memory_allocation_exception) {
00716      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00717 std::endl;
00718      std::cerr << memory_allocation_exception.what() << std::endl;
00719    }
00720    memset(prem_apps_,
00721           mtk::kZero,
00722           sizeof(prem_apps_[0])*num_bndy_coeffs_*num_bndy_approxs_);
00723
00725
00726    for (auto ll = 0; ll < num_bndy_approxs_; ++ll) {
00727
00728      // Re-check new generator vector for every iteration except for the first.
00729      #if MTK_DEBUG_LEVEL > 0
00730      if (ll > 0) {
00731        std::cout << "gg" << ll << " =" << std::endl;
00732        for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00733          std::cout << std::setw(12) << gg[ii];
00734        }
00735        std::cout << std::endl << std::endl;
00736      }
00737      #endif
00738
00740
00741      bool transpose{false};
00742
00743      mtk::DenseMatrix aa(gg,
00744                          num_bndy_coeffs_, order_accuracy_ + 1,
00745                          transpose);
00746
00747      #if MTK_DEBUG_LEVEL > 0
00748      std::cout << "aa_" << ll << " =" << std::endl;
00749      std::cout << aa << std::endl;
00750      #endif
00751
00753
00754      mtk::Real *ob{};
00755
00756      auto ob_ld = num_bndy_coeffs_;
00757
00758      try {
00759        ob = new mtk::Real[ob_ld];
00760      } catch (std::bad_alloc &memory_allocation_exception) {
00761        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00762          std::endl;
00763        std::cerr << memory_allocation_exception.what() << std::endl;
00764      }
00765      memset(ob, mtk::kZero, sizeof(ob[0])*ob_ld);
00766
00767      ob[1] = mtk::kOne;
00768
00769      #if MTK_DEBUG_LEVEL > 0
00770      std::cout << "ob = " << std::endl << std::endl;
00771      for (auto ii = 0; ii < ob_ld; ++ii) {
00772        std::cout << std::setw(12) << ob[ii] << std::endl;
00773      }
00774      std::cout << std::endl;
00775      #endif
00776
00778
00779      // However, this is an under-determined system of equations. So we can not
00780      // use the same LAPACK routine (dgesv_). We will instead use dgels_, through
00781      // our LAPACKAdapter class.
00782
00783      int info_{
```

```
00784        mtk::LAPACKAdapter::SolveRectangularDenseSystem(aa, ob
      , ob_ld)};
00785
00786      #if MTK_DEBUG_LEVEL > 0
00787      if (!info_) {
00788        std::cout << "System successfully solved!" << std::endl << std::endl;
00789      } else {
00790        std::cerr << "Error solving system! info = " << info_ << std::endl;
00791      }
00792      #endif
00793
00794      #if MTK_DEBUG_LEVEL > 0
00795      std::cout << "ob =" << std::endl;
00796      for (auto ii = 0; ii < ob_ld; ++ii) {
00797        std::cout << std::setw(12) << ob[ii] << std::endl;
00798      }
00799      std::cout << std::endl;
00800      #endif
00801
00802
00803
00804      // This implies a DAXPY operation. However, we must construct the arguments
00805      // for this operation.
00806
00807      // Save them into the ob_bottom array:
00808
00809
00810      Real *ob_bottom{}; // Bottom part of the attained kernel used to scale it.
00811
00812      try {
00813        ob_bottom = new mtk::Real[dim_null_];
00814      } catch (std::bad_alloc &memory_allocation_exception) {
00815        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00816          std::endl;
00817        std::cerr << memory_allocation_exception.what() << std::endl;
00818      }
00819      memset(ob_bottom, mtk::kZero, sizeof(ob_bottom[0])*dim_null_);
00820
00821      for (auto ii = 0; ii < dim_null_; ++ii) {
00822        ob_bottom[(dim_null_ - 1) - ii] = ob[num_bndy_coeffs_ - ii - 1];
00823      }
00824
00825      #if MTK_DEBUG_LEVEL > 0
00826      std::cout << "ob_bottom =" << std::endl;
00827      for (auto ii = 0; ii < dim_null_; ++ii) {
00828        std::cout << std::setw(12) << ob_bottom[ii] << std::endl;
00829      }
00830      std::cout << std::endl;
00831      #endif
00832
00833
00834
00835      // We must computed an scaled ob, sob, using the scaled null-space in
00836      // rat_basis_null_space_.
00837      // Such operation is: sob = ob - rat_basis_null_space_*ob_bottom
00838      // or:                 ob = -1.0*rat_basis_null_space_*ob_bottom + 1.0*ob
00839      // thus:               Y =    a*A    *x          +   b*Y (DAXPY).
00840
00841      #if MTK_DEBUG_LEVEL > 0
00842      std::cout << "Rational basis for the null-space:" << std::endl;
00843      std::cout << rat_basis_null_space_ << std::endl;
00844      #endif
00845
00846      mtk::Real alpha{-mtk::kOne};
00847      mtk::Real beta{mtk::kOne};
00848
00849      mtk::BLASAdapter::RealDenseMV(alpha, rat_basis_null_space_,
00850                                    ob_bottom, beta, ob);
00851
00852      #if MTK_DEBUG_LEVEL > 0
00853      std::cout << "scaled ob:" << std::endl;
00854      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00855        std::cout << std::setw(12) << ob[ii] << std::endl;
00856      }
00857      std::cout << std::endl;
00858      #endif
00859
00860      // We save the recently scaled solution, into an array containing these.
00861      // We can NOT start building the pi matrix, simply because I want that part
00862      // to be separated since its construction depends on the algorithm we want
00863      // to implement.
00864
00865      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00866        prem_apps_[ii*num_bndy_approxs_ + ll] = ob[ii];
```

```
00867      }
00868
00869      // After the first iteration, simply shift the entries of the last
00870      // generator vector used:
00871      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00872        gg[ii]--;
00873      }
00874
00875      // Garbage collection for this loop:
00876      delete[] ob;
00877      ob = nullptr;
00878
00879      delete[] ob_bottom;
00880      ob_bottom = nullptr;
00881    } // End of: for (ll = 0; ll < dim_null; ll++);
00882
00883    #if MTK_DEBUG_LEVEL > 0
00884    std::cout << "Matrix post-scaled preliminary apps: " << std::endl;
00885    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00886      for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
00887        std::cout << std::setw(12) << prem_apps_[ii*num_bndy_approxs_ + jj];
00888      }
00889      std::cout << std::endl;
00890    }
00891    std::cout << std::endl;
00892    #endif
00893
00894    delete[] gg;
00895    gg = nullptr;
00896
00897    return true;
00898 }
00899
00900 bool mtk::Grad1D::ComputeWeights() {
00901
00902    // Matrix to copmpute the weights as in the CRSA.
00903    mtk::DenseMatrix pi(num_bndy_coeffs_, num_bndy_coeffs_ - 1);
00904
00906
00907    // Assemble the pi matrix using:
00908    // 1. The collection of scaled preliminary approximations.
00909    // 2. The collection of coefficients approximating at the interior.
00910    // 3. The scaled basis for the null-space.
00911
00912    // 1.1. Process array of scaled preliminary approximations.
00913
00914    // These are queued in scaled_solutions. Each one of these, will be a column
00915    // of the pi matrix:
00916    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00917      for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
00918        pi.data()[ii*(2*(num_bndy_approxs_ - 1) + (order_accuracy_/2 + 1)) + jj] =
00919          prem_apps_[ii*num_bndy_approxs_ + jj];
00920      }
00921    }
00922
00923    // 1.2. Add columns from known stencil approximating at the interior.
00924
00925    // However, these must be padded by zeros, according to their position in the
00926    // final pi matrix:
00927    auto mm = 1;
00928    for (auto jj = num_bndy_approxs_; jj < order_accuracy_; ++jj) {
00929      for (auto ii = 0; ii < order_accuracy_; ++ii) {
00930        auto de = (ii + mm)*(2*(num_bndy_approxs_ - 1) +
00931          (order_accuracy_/2 + 1)) + jj;
00932        pi.data()[de] = coeffs_interior_[ii];
00933      }
00934      ++mm;
00935    }
00936
00937    rat_basis_null_space_.OrderColMajor();
00938
00939    #if MTK_DEBUG_LEVEL > 0
00940    std::cout << "Rational basis for the null-space (col. major):" << std::endl;
00941    std::cout << rat_basis_null_space_ << std::endl;
00942    #endif
00943
00944    // 1.3. Add final set of columns: rational basis for null-space.
00945    for (auto jj = dim_null_ + (order_accuracy_/2 + 1);
00946         jj < num_bndy_coeffs_ - 1; ++jj) {
00947      for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
00948        auto og =
```

```
00949            (jj - (dim_null_ + (order_accuracy_/2 + 1)))*num_bndy_coeffs_ + ii;
00950          auto de = ii*(2*dim_null_ + (order_accuracy_/2 + 1)) + jj;
00951          pi.data()[de] = rat_basis_null_space_.data()[og];
00952        }
00953      }
00954
00955      #if MTK_DEBUG_LEVEL >0
00956      std::cout << "coeffs_interior_ =" << std::endl;
00957      for (auto ii = 0; ii < order_accuracy_; ++ii) {
00958        std::cout << std::setw(12) << coeffs_interior_[ii];
00959      }
00960      std::cout << std::endl << std::endl;
00961      #endif
00962
00963      #if MTK_DEBUG_LEVEL >0
00964      std::cout << "Constructed pi matrix for CRS Algorithm: " << std::endl;
00965      std::cout << pi << std::endl;
00966      #endif
00967
00968
00969
00970      // This imposes the mimetic condition.
00971
00972      mtk::Real *hh{};  // Right-hand side to compute weights in the C{R,B}SA.
00973
00974      try {
00975        hh = new mtk::Real[num_bndy_coeffs_];
00976      } catch (std::bad_alloc &memory_allocation_exception) {
00977        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00978          std::endl;
00979        std::cerr << memory_allocation_exception.what() << std::endl;
00980      }
00981      memset(hh, mtk::kZero, sizeof(hh[0])*num_bndy_coeffs_);
00982
00983      hh[0] = -mtk::kOne;
00984      for (auto ii = (order_accuracy_/2 + 2 - 1); ii < num_bndy_coeffs_; ++ii) {
00985        auto aux_xx = mtk::kZero;
00986        for (auto jj = 0; jj < ((ii - (order_accuracy_/2 - 1)) - 1); ++jj) {
00987          aux_xx += coeffs_interior_[jj];
00988        }
00989        hh[ii] = -mtk::kOne*aux_xx;
00990      }
00991
00992
00993
00994      // That is, we construct a system, to solve for the weights.
00995
00996      // Once again we face the challenge of solving with LAPACK. However, for the
00997      // CRSA, this matrix PI is over-determined, since it has more rows than
00998      // unknowns. However, according to the theory, the solution to this system is
00999      // unique. We will use dgels_.
01000
01001      try {
01002        weights_cbs_ = new mtk::Real[num_bndy_coeffs_];
01003      } catch (std::bad_alloc &memory_allocation_exception) {
01004        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01005          std::endl;
01006        std::cerr << memory_allocation_exception.what() << std::endl;
01007      }
01008      memset(weights_cbs_, mtk::kZero, sizeof(weights_cbs_[0])*num_bndy_coeffs_);
01009
01010      int weights_ld{pi.num_cols() + 1};
01011
01012      // Preserve hh.
01013      std::copy(hh, hh + weights_ld, weights_cbs_);
01014
01015      pi.Transpose();
01016
01017      int info{
01018        mtk::LAPACKAdapter::SolveRectangularDenseSystem(pi,
01019                                                        weights_cbs_, weights_ld)
01020      };
01021
01022      #if MTK_DEBUG_LEVEL > 0
01023      if (!info) {
01024        std::cout << "System successfully solved!" << std::endl << std::endl;
01025      } else {
01026        std::cerr << "Error solving system! info = " << info << std::endl;
01027      }
01028      #endif
01029
01030      #if MTK_DEBUG_LEVEL > 0
01031      std::cout << "hh =" << std::endl;
```

```
01032    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01033      std::cout << std::setw(11) << hh[ii] << std::endl;
01034    }
01035    std::cout << std::endl;
01036    #endif
01037
01038    // Preserve the original weights for research.
01039
01040    try {
01041      weights_crs_ = new mtk::Real[num_bndy_coeffs_];
01042    } catch (std::bad_alloc &memory_allocation_exception) {
01043      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01044        std::endl;
01045      std::cerr << memory_allocation_exception.what() << std::endl;
01046    }
01047    memset(weights_crs_, mtk::kZero, sizeof(weights_crs_[0])*num_bndy_coeffs_);
01048
01049    std::copy(weights_cbs_, weights_cbs_ + (weights_ld - 1), weights_crs_);
01050
01051    #if MTK_DEBUG_LEVEL > 0
01052    std::cout << "weights_CRSA + lambda =" << std::endl;
01053    for (auto ii = 0; ii < weights_ld - 1; ++ii) {
01054      std::cout << std::setw(12) << weights_crs_[ii] << std::endl;
01055    }
01056    std::cout << std::endl;
01057    #endif
01058
01059    if (order_accuracy_ >= mtk::kCriticalOrderAccuracyGrad) {
01060
01061
01062      int minrow_{std::numeric_limits<int>::infinity()};
01063
01064      mtk::Real norm{mtk::BLASAdapter::RealNRM2(weights_cbs_,
01065   order_accuracy_)};
01065      mtk::Real minnorm{std::numeric_limits<mtk::Real>::infinity()};
01066
01067
01068
01069      mtk::DenseMatrix phi(order_accuracy_ + 1, order_accuracy_);
01070
01071      // 6.1. Insert preliminary approximations to first set of columns.
01072
01073      for (auto ii = 0; ii < order_accuracy_ + 1; ++ii) {
01074        for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01075          phi.data()[ii*(order_accuracy_) + jj] =
01076            prem_apps_[ii*num_bndy_approxs_ + jj];
01077        }
01078      }
01079
01080      // 6.2. Skip a column and negate preliminary approximations.
01081
01082      for (auto jj = 0; jj < order_accuracy_ + 1; jj++) {
01083        for (auto ii = 1; ii < num_bndy_approxs_; ii++) {
01084          auto de = (ii+ order_accuracy_ - num_bndy_approxs_+ jj*order_accuracy_);
01085          auto og = (num_bndy_approxs_ - ii + (jj)*num_bndy_approxs_);
01086          phi.data()[de] = -prem_apps_[og];
01087        }
01088      }
01089
01090      // 6.3. Flip negative columns up-down.
01091
01092      for (auto ii = 0; ii < order_accuracy_/2; ii++) {
01093        for (auto jj = num_bndy_approxs_ + 1; jj < order_accuracy_; jj++) {
01094          auto aux = phi.data()[ii*order_accuracy_ + jj];
01095          phi.data()[ii*order_accuracy_ + jj] =
01096            phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj];
01097          phi.data()[(order_accuracy_ - ii)*order_accuracy_ + jj] = aux;
01098        }
01099      }
01100
01101      // 6.4. Insert stencil.
01102
01103      auto mm = 0;
01104      for (auto jj = num_bndy_approxs_; jj < num_bndy_approxs_ +  1; jj++) {
01105        for (auto ii = 0; ii < order_accuracy_ + 1; ii++) {
01106          if (ii == 0) {
01107            phi.data()[jj] = 0.0;
01108          } else {
01109            phi.data()[(ii + mm)*order_accuracy_ + jj] = coeffs_interior_[ii - 1];
01110          }
01111        }
01112        mm++;
01113      }
```

```
01114
01115       #if MTK_DEBUG_LEVEL > 0
01116       std::cout << "phi =" << std::endl;
01117       std::cout << phi << std::endl;
01118       #endif
01119
01120
01121
01122       mtk::Real *lamed{};  // Used to build big lambda.
01123
01124       try {
01125         lamed = new mtk::Real[num_bndy_approxs_ - 1];
01126       } catch (std::bad_alloc &memory_allocation_exception) {
01127         std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01128           std::endl;
01129         std::cerr << memory_allocation_exception.what() << std::endl;
01130       }
01131       memset(lamed, mtk::kZero, sizeof(lamed[0])*(num_bndy_approxs_ - 1));
01132
01133       for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01134         lamed[ii] = hh[ii + order_accuracy_ + 1] ;
01135       }
01136
01137       #if MTK_DEBUG_LEVEL > 0
01138       std::cout << "lamed =" << std::endl;
01139       for (auto ii = 0; ii < num_bndy_approxs_ - 1; ++ii) {
01140         std::cout << std::setw(12) << lamed[ii] << std::endl;
01141       }
01142       std::cout << std::endl;
01143       #endif
01144
01145       for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01146         mtk::Real temp = mtk::kZero;
01147         for(auto jj = 0; jj < num_bndy_approxs_ - 1; ++jj) {
01148           temp = temp +
01149             lamed[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01150         }
01151         hh[ii] = hh[ii] - temp;
01152       }
01153
01154       #if MTK_DEBUG_LEVEL > 0
01155       std::cout << "big_lambda =" << std::endl;
01156       for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01157         std::cout << std::setw(12) << hh[ii] << std::endl;
01158       }
01159       std::cout << std::endl;
01160       #endif
01161
01163
01164       int copy_result{};  // Should I replace the solution... not for now.
01165
01166       mtk::Real normerr_; // Norm of the error for the solution on each row.
01167
01168       for(auto row_= 0; row_ < order_accuracy_ + 1; ++row_) {
01169         normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
    data(),
01170                                                            order_accuracy_ + 1,
01171                                                            order_accuracy_,
01172                                                            order_accuracy_,
01173                                                            hh,
01174                                                            weights_cbs_,
01175                                                            row_,
01176                                                            mimetic_threshold_,
01177                                                            copy_result);
01178         mtk::Real aux{normerr_/norm};
01179
01180         #if MTK_DEBUG_LEVEL>0
01181         std::cout << "Relative norm: " << aux << " " << std::endl;
01182         std::cout << std::endl;
01183         #endif
01184
01185         if (aux < minnorm) {
01186           minnorm = aux;
01187           minrow_= row_;
01188         }
01189       }
01190
01191       #if MTK_DEBUG_LEVEL > 0
01192       std::cout << "weights_CBSA + lambda (after brute force search):" <<
01193         std::endl;
01194       for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01195         std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
```

```
01196      }
01197      std::cout << std::endl;
01198      #endif
01199
01201
01202      // After we know which row yields the smallest relative norm that row is
01203      // chosen to be the objective function and the result of the optimizer is
01204      // chosen to be the new weights_.
01205
01206      #if MTK_DEBUG_LEVEL > 0
01207      std::cout << "Minimum Relative Norm " << minnorm << " found at row " <<
01208        minrow_ + 1 << std::endl;
01209      std::cout << std::endl;
01210      #endif
01211
01212      copy_result = 1;
01213      normerr_ = mtk::GLPKAdapter::SolveSimplexAndCompare(phi.
    data(),
01214                                                        order_accuracy_ + 1,
01215                                                        order_accuracy_,
01216                                                        order_accuracy_,
01217                                                        hh,
01218                                                        weights_cbs_,
01219                                                        minrow_,
01220                                                        mimetic_threshold_,
01221                                                        copy_result);
01222      mtk::Real aux_{normerr_/norm};
01223      #if MTK_DEBUG_LEVEL > 0
01224      std::cout << "Relative norm: " << aux_ << std::endl;
01225      std::cout << std::endl;
01226      #endif
01227
01228      delete [] lamed;
01229      lamed = nullptr;
01230    }
01231
01232    delete [] hh;
01233    hh = nullptr;
01234
01235    return true;
01236 }
01237
01238 bool mtk::Grad1D::ComputeStencilBoundaryGrid(void) {
01239
01240    #if MTK_DEBUG_LEVEL > 0
01241    std::cout << "weights_* + lambda =" << std::endl;
01242    for (auto ii = 0; ii < num_bndy_coeffs_ - 1; ++ii) {
01243      std::cout << std::setw(12) << weights_cbs_[ii] << std::endl;
01244    }
01245    std::cout << std::endl;
01246    #endif
01247
01249
01250    mtk::Real *lambda{}; // Collection of bottom values from weights_.
01251
01252    try {
01253      lambda = new mtk::Real[dim_null_];
01254    } catch (std::bad_alloc &memory_allocation_exception) {
01255      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01256        std::endl;
01257      std::cerr << memory_allocation_exception.what() << std::endl;
01258    }
01259    memset(lambda, mtk::kZero, sizeof(lambda[0])*dim_null_);
01260
01261    for (auto ii = 0; ii < dim_null_; ++ii) {
01262      lambda[ii] = weights_cbs_[order_accuracy_ + ii];
01263    }
01264
01265    #if MTK_DEBUG_LEVEL > 0
01266    std::cout << "lambda =" << std::endl;
01267    for (auto ii = 0; ii < dim_null_; ++ii) {
01268      std::cout << std::setw(12) << lambda[ii] << std::endl;
01269    }
01270    std::cout << std::endl;
01271    #endif
01272
01274
01275    mtk::Real *alpha{}; // Collection of alpha values.
01276
01277    try {
01278      alpha = new mtk::Real[dim_null_];
```

```
01279    } catch (std::bad_alloc &memory_allocation_exception) {
01280      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01281        std::endl;
01282      std::cerr << memory_allocation_exception.what() << std::endl;
01283    }
01284    memset(alpha, mtk::kZero, sizeof(alpha[0])*dim_null_);
01285
01286    for (auto ii = 0; ii < dim_null_; ++ii) {
01287      alpha[ii] = lambda[ii]/weights_cbs_[ii] ;
01288    }
01289
01290    #if MTK_DEBUG_LEVEL > 0
01291    std::cout << "alpha =" << std::endl;
01292    for (auto ii = 0; ii < dim_null_; ++ii) {
01293      std::cout << std::setw(12) << alpha[ii] << std::endl;
01294    }
01295    std::cout << std::endl;
01296    #endif
01297
01298
01299
01300    try {
01301      mim_bndy_ = new mtk::Real[num_bndy_coeffs_*num_bndy_approxs_];
01302    } catch (std::bad_alloc &memory_allocation_exception) {
01303      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01304        std::endl;
01305      std::cerr << memory_allocation_exception.what() << std::endl;
01306    }
01307    memset(mim_bndy_,
01308           mtk::kZero,
01309           sizeof(mim_bndy_[0])*num_bndy_coeffs_*num_bndy_approxs_);
01310
01311    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01312      for (auto jj = 0; jj < (num_bndy_approxs_ - 1); ++jj) {
01313        mim_bndy_[ii*num_bndy_approxs_ + jj] =
01314          prem_apps_[ii*num_bndy_approxs_ + jj] +
01315          alpha[jj]*rat_basis_null_space_.data()[jj*num_bndy_coeffs_ + ii];
01316      }
01317    }
01318
01319    for(auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01320      mim_bndy_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)] =
01321        prem_apps_[ii*num_bndy_approxs_ + (num_bndy_approxs_ - 1)];
01322    }
01323
01324    #if MTK_DEBUG_LEVEL > 0
01325    std::cout << "Collection of mimetic approximations:" << std::endl;
01326    for (auto ii = 0; ii < num_bndy_coeffs_; ++ii) {
01327      for (auto jj = 0; jj < num_bndy_approxs_; ++jj) {
01328        std::cout << std::setw(13) << mim_bndy_[ii*num_bndy_approxs_ + jj];
01329      }
01330      std::cout << std::endl;
01331    }
01332    std::cout << std::endl;
01333    #endif
01334
01335    delete[] lambda;
01336    lambda = nullptr;
01337
01338    delete[] alpha;
01339    alpha = nullptr;
01340
01341    return true;
01342  }
01343
01344  bool mtk::Grad1D::AssembleOperator(void) {
01345
01346    // The output array will have this form:
01347    // 1. The first entry of the array will contain the used order kk.
01348    // 2. The second entry of the array will contain the collection of
01349    // approximating coefficients for the interior of the grid.
01350    // 3. The third entry will contain a collection of weights.
01351    // 4. The next dim_null - 1 entries will contain the collections of
01352    // approximating coefficients for the west boundary of the grid.
01353
01354    gradient_length_ = 1 + order_accuracy_ + order_accuracy_ +
01355      num_bndy_approxs_*num_bndy_coeffs_;
01356
01357    #if MTK_DEBUG_LEVEL > 0
01358    std::cout << "gradient_length_ = " << gradient_length_ << std::endl;
01359    #endif
01360
```

```
01361      try {
01362        gradient_ = new mtk::Real[gradient_length_];
01363      } catch (std::bad_alloc &memory_allocation_exception) {
01364        std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
01365          std::endl;
01366        std::cerr << memory_allocation_exception.what() << std::endl;
01367      }
01368      memset(gradient_, mtk::kZero, sizeof(gradient_[0])*gradient_length_);
01369
01371
01372      gradient_[0] = order_accuracy_;
01373
01376
01377      for (auto ii = 0; ii < order_accuracy_; ++ii) {
01378        gradient_[ii + 1] = coeffs_interior_[ii];
01379      }
01380
01382
01383      for (auto ii = 0; ii < order_accuracy_; ++ii) {
01384        gradient_[(order_accuracy_ + 1) + ii] = weights_cbs_[ii];
01385      }
01386
01389
01390      int offset{2*order_accuracy_ + 1};
01391
01392      int aux {}; // Auxiliary variable.
01393
01394      if (order_accuracy_ > mtk::kDefaultOrderAccuracy) {
01395        for (auto ii = 0; ii < num_bndy_approxs_ ; ii++) {
01396          for (auto jj = 0; jj < num_bndy_coeffs_; jj++) {
01397            gradient_[offset + aux] = mim_bndy_[jj*num_bndy_approxs_ + ii];
01398            aux++;
01399          }
01400        }
01401      } else {
01402        gradient_[offset + 0] = prem_apps_[0];
01403        gradient_[offset + 1] = prem_apps_[1];
01404        gradient_[offset + 2] = prem_apps_[2];
01405      }
01406
01407      #if MTK_DEBUG_LEVEL > 0
01408      std::cout << "1D " << order_accuracy_ << "-order grad built!" << std::endl;
01409      std::cout << std::endl;
01410      #endif
01411
01412      return true;
01413 }
```

17.45 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:
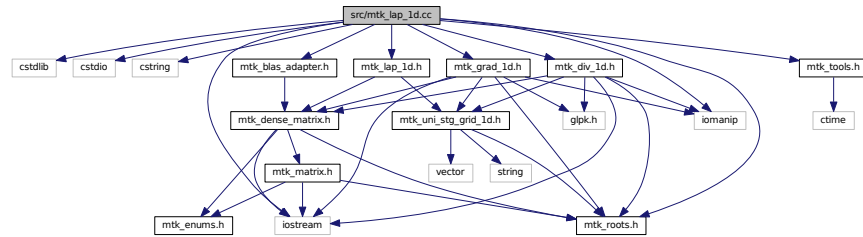
Namespaces

- **mtk**

    Mimetic Methods Toolkit namespace.

Functions

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

17.45.1   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.

17.46   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. Documentation related to said modifications should be included.
00021
00022 2. Redistributions of source code must be done through direct
00023 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00024
00025 3. Redistributions of source code must retain the above copyright notice, this
00026 list of conditions and the following disclaimer.
00027
00028 4. Redistributions in binary form must reproduce the above copyright notice,
00029 this list of conditions and the following disclaimer in the documentation and/or
00030 other materials provided with the distribution.
00031
```

```
00032 5. Usage of the binary form on proprietary applications shall require explicit
00033 prior written permission from the the copyright holders.
00034
00035 6. Neither the name of the copyright holder nor the names of its contributors
00036 may be used to endorse or promote products derived from this software without
00037 specific prior written permission.
00038
00039 The copyright holders provide no reassurances that the source code provided does
00040 not infringe any patent, copyright, or any other intellectual property rights of
00041 third parties. The copyright holders disclaim any liability to any recipient for
00042 claims brought against recipient by any third party for infringement of that
00043 parties intellectual property rights.
00044
00045 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00046 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00047 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00048 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00049 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00050 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00051 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00052 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00053 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00054 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00055 */
00056
00057 #include <cstdlib>
00058 #include <cstdio>
00059 #include <cstring>
00060
00061 #include <iostream>
00062 #include <iomanip>
00063
00064 #include "mtk_roots.h"
00065 #include "mtk_tools.h"
00066 #include "mtk_blas_adapter.h"
00067 #include "mtk_grad_1d.h"
00068 #include "mtk_div_1d.h"
00069 #include "mtk_lap_1d.h"
00070
00071 namespace mtk {
00072
00073 std::ostream& operator <<(std::ostream &stream, mtk::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   mimetic_threshold_(mtk::kDefaultMimeticThreshold) {}
00112
00113 mtk::Lap1D::~Lap1D() {
00114
00115   delete [] laplacian_;
```

```
00116    laplacian_ = nullptr;
00117 }
00118
00119 bool mtk::Lap1D::ConstructLap1D(int order_accuracy,
00120                                 mtk::Real mimetic_threshold) {
00121
00122    #if MTK_DEBUG_LEVEL > 0
00123    mtk::Tools::Prevent(order_accuracy < 2, __FILE__, __LINE__, __func__);
00124    mtk::Tools::Prevent((order_accuracy%2) != 0, __FILE__, __LINE__, __func__);
00125    mtk::Tools::Prevent(mimetic_threshold <= mtk::kZero,
00126                        __FILE__, __LINE__, __func__);
00127
00128    if (order_accuracy >= mtk::kCriticalOrderAccuracyDiv) {
00129      std::cout << "WARNING: Numerical accuracy is high." << std::endl;
00130    }
00131
00132    std::cout << "order_accuracy_ = " << order_accuracy << std::endl;
00133    std::cout << "mimetic_threshold_ = " << mimetic_threshold << std::endl;
00134    #endif
00135
00136    order_accuracy_ = order_accuracy;
00137    mimetic_threshold_ = mimetic_threshold;
00138
00140    mtk::Grad1D grad; // Mimetic gradient.
00141
00142
00143    bool info = grad.ConstructGrad1D(order_accuracy_, mimetic_threshold_);
00144
00145    if (!info) {
00146      std::cerr << "Mimetic grad could not be built." << std::endl;
00147      return false;
00148    }
00149
00151
00152    mtk::Div1D div; // Mimetic divergence.
00153
00154    info = div.ConstructDiv1D(order_accuracy_, mimetic_threshold_);
00155
00156    if (!info) {
00157      std::cerr << "Mimetic div could not be built." << std::endl;
00158      return false;
00159    }
00160
00162
00163    // Since these are mimetic operator, we must multiply the matrices arising
00164    // from both the divergence and the Laplacian, in order to get the
00165    // approximating coefficients for the Laplacian operator.
00166
00167    // However, we must choose a grid that implied a step size of 1, so to get
00168    // the approximating coefficients, without being affected from the
00169    // normalization with respect to the grid.
00170
00171    // Also, the grid must be of the minimum size to support the requested order
00172    // of accuracy. We must please the divergence.
00173
00174    mtk::UniStgGrid1D aux(mtk::kZero,
00175                          (mtk::Real) 3*order_accuracy_ - 1,
00176                          3*order_accuracy_ - 1);
00177
00178    #if MTK_DEBUG_LEVEL > 0
00179    std::cout << "aux =" << std::endl;
00180    std::cout << aux << std::endl;
00181    std::cout <<"aux.delta_x() = " << aux.delta_x() << std::endl;
00182    std::cout << std::endl;
00183    #endif
00184
00185    mtk::DenseMatrix grad_m(grad.ReturnAsDenseMatrix(aux));
00186
00187    #if MTK_DEBUG_LEVEL > 0
00188    std::cout << "grad_m =" << std::endl;
00189    std::cout << grad_m << std::endl;
00190    #endif
00191
00192    mtk::DenseMatrix div_m(div.ReturnAsDenseMatrix(aux));
00193
00194    #if MTK_DEBUG_LEVEL > 0
00195    std::cout << "div_m =" << std::endl;
00196    std::cout << div_m << std::endl;
00197    #endif
00198
00202
```

```
00203    mtk::DenseMatrix lap; // Laplacian matrix to hold to computed coefficients.
00204
00205    lap = mtk::BLASAdapter::RealDenseMM(div_m, grad_m);
00206
00207    #if MTK_DEBUG_LEVEL > 0
00208    std::cout << "lap =" << std::endl;
00209    std::cout << lap << std::endl;
00210    #endif
00211
00213
00215
00216    // The output array will have this form:
00217    // 1. The first entry of the array will contain the used order kk.
00218    // 2. The second entry of the array will contain the collection of
00219    // approximating coefficients for the interior of the grid.
00220    // 3. The next entries will contain the collections of approximating
00221    // coefficients for the west boundary of the grid.
00222
00223    laplacian_length_ = 1 + (2*order_accuracy_ - 1) +
00224      (order_accuracy_ - 1)*(2*order_accuracy_);
00225
00226    #if MTK_DEBUG_LEVEL > 0
00227    std::cout << "laplacian_length_ = " << laplacian_length_ << std::endl;
00228    std::cout << std::endl;
00229    #endif
00230
00231    try {
00232      laplacian_ = new mtk::Real[laplacian_length_];
00233    } catch (std::bad_alloc &memory_allocation_exception) {
00234      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00235        std::endl;
00236      std::cerr << memory_allocation_exception.what() << std::endl;
00237    }
00238    memset(laplacian_, mtk::kZero, sizeof(laplacian_[0])*laplacian_length_);
00239
00241
00242    laplacian_[0] = order_accuracy_;
00243
00246
00247    for (auto ii = 0; ii < 2*order_accuracy_ - 1; ++ii) {
00248      laplacian_[ii + 1] = lap.GetValue(1 + (order_accuracy_ - 1), ii + 1);
00249    }
00250
00252
00253    auto offset = 1 + (2*order_accuracy_ - 1);
00254
00255    for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00256      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00257        laplacian_[offset + ii*(2*order_accuracy_) + jj] =
00258          lap.GetValue(1 + ii,jj);
00259      }
00260    }
00261
00262    return true;
00263 }
00264
00265 mtk::DenseMatrix mtk::Lap1D::ReturnAsDenseMatrix(const
       UniStgGrid1D &grid) {
00266
00267    int nn{grid.num_cells_x()};  // Number of cells on the grid.
00268
00269    #if MTK_DEBUG_LEVEL > 0
00270    mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00271    mtk::Tools::Prevent(nn < 3*order_accuracy_ - 1, __FILE__, __LINE__, __func__);
00272    #endif
00273
00274    mtk::DenseMatrix lap(nn + 2, nn + 2); // Laplacian matrix to be returned.
00275
00276    mtk::Real idx{mtk::kOne/(grid.delta_x()*grid.delta_x())}; // Inverse of
       dx^2.
00277
00279
00280    auto offset = (1 + 2*order_accuracy_ - 1);
00281
00282    for (auto ii = 0; ii < order_accuracy_ - 1; ++ii) {
00283      for (auto jj = 0; jj < 2*order_accuracy_; ++jj) {
00284        lap.SetValue(1 + ii,
00285                     jj,
00286                     idx*laplacian_[offset + ii*2*order_accuracy_ + jj]);
00287      }
00288    }
```

```
00289
00291
00292    offset = 1 + (order_accuracy_ - 1);
00293
00294    int kk{1};
00295    for (auto ii = order_accuracy_; ii <= nn - (order_accuracy_ - 1); ++ii) {
00296      int mm{1};
00297      for (auto jj = 0; jj < 2*order_accuracy_ - 1; ++jj) {
00298        lap.SetValue(ii, jj + kk, idx*laplacian_[mm]);
00299        mm = mm + 1;
00300      }
00301      kk = kk + 1;
00302    }
00303
00305
00306    offset = (1 + 2*order_accuracy_ - 1);
00307
00308    auto aux = order_accuracy_ + (nn - 2*(order_accuracy_ - 1));
00309
00310    auto ll = 1;
00311    auto rr = 1;
00312    for (auto ii = nn; ii > aux - 1; --ii) {
00313      auto cc = 0;
00314      for (auto jj = nn + 2 - 1; jj >= (nn + 2) - 2*order_accuracy_; --jj) {
00315        lap.SetValue(ii, jj, lap.GetValue(rr,cc));
00316        ++ll;
00317        ++cc;
00318      }
00319      rr++;
00320    }
00321
00328
00329    return lap;
00330  }
00331
00332 mtk::Real* mtk::Lap1D::Data(const UniStgGrid1D &grid) {
00333
00334    mtk::DenseMatrix tmp;
00335
00336    tmp = ReturnAsDenseMatrix(grid);
00337
00338    return tmp.data();
00339  }
```

17.47 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
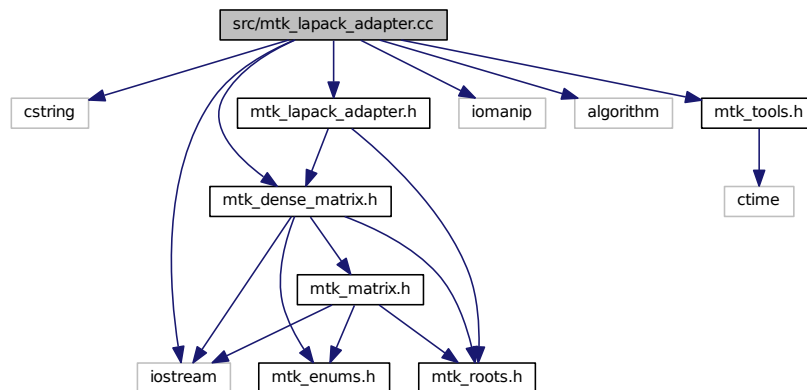
  Mimetic Methods Toolkit namespace.

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.

17.47.1 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** is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

See Also

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

Author

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

Definition in file mtk_lapack_adapter.cc.

17.48 mtk_lapack_adapter.cc

```
00001
00018 /*
00019 Copyright (C) 2015, Computational Science Research Center, San Diego State
00020 University. All rights reserved.
00021
00022 Redistribution and use in source and binary forms, with or without modification,
00023 are permitted provided that the following conditions are met:
00024
00025 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00026 and a copy of the modified files should be reported once modifications are
00027 completed. Documentation related to said modifications should be included.
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 of source code must retain the above copyright notice, this
00033 list of conditions and the following disclaimer.
00034
00035 4. 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 5. Usage of the binary form on proprietary applications shall require explicit
00040 prior written permission from the the copyright holders.
00041
00042 6. Neither the name of the copyright holder nor the names of its contributors
00043 may be used to endorse or promote products derived from this software without
00044 specific prior written permission.
00045
00046 The copyright holders provide no reassurances that the source code provided does
00047 not infringe any patent, copyright, or any other intellectual property rights of
00048 third parties. The copyright holders disclaim any liability to any recipient for
00049 claims brought against recipient by any third party for infringement of that
00050 parties intellectual property rights.
00051
00052 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00053 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00054 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00055 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00056 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00057 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00058 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00059 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00060 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00061 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00062 */
00063
00064 #include <cstring>
00065
00066 #include <iostream>
00067 #include <iomanip>
00068
00069 #include <algorithm>
00070
00071 #include "mtk_tools.h"
00072 #include "mtk_dense_matrix.h"
00073 #include "mtk_lapack_adapter.h"
00074
00075 namespace mtk {
00076
00077 extern "C" {
00078
00079 #ifdef MTK_PRECISION_DOUBLE
00080
00099 void dgesv_(int* n,
00100             int* nrhs,
00101             Real* a,
00102             int* lda,
```

```
00103              int* ipiv,
00104              Real* b,
00105              int* ldb,
00106              int* info);
00107 #else
00108
00127 void sgesv_(int* n,
00128              int* nrhs,
00129              Real* a,
00130              int* lda,
00131              int* ipiv,
00132              Real* b,
00133              int* ldb,
00134              int* info);
00135 #endif
00136
00137 #ifdef MTK_PRECISION_DOUBLE
00138
00181 void dgels_(char* trans,
00182              int* m,
00183              int* n,
00184              int* nrhs,
00185              Real* a,
00186              int* lda,
00187              Real* b,
00188              int* ldb,
00189              Real* work,
00190              int* lwork,
00191              int* info);
00192 #else
00193
00236 void sgels_(char* trans,
00237              int* m,
00238              int* n,
00239              int* nrhs,
00240              Real* a,
00241              int* lda,
00242              Real* b,
00243              int* ldb,
00244              Real* work,
00245              int* lwork,
00246              int* info);
00247 #endif
00248
00249 #ifdef MTK_PRECISION_DOUBLE
00250
00279 void dgeqrf_(int *m,
00280              int *n,
00281              Real *a,
00282              int *lda,
00283              Real *tau,
00284              Real *work,
00285              int *lwork,
00286              int *info);
00287 #else
00288
00317 void sgeqrf_(int *m,
00318              int *n,
00319              Real *a,
00320              int *lda,
00321              Real *tau,
00322              Real *work,
00323              int *lwork,
00324              int *info);
00325 #endif
00326
00327 #ifdef MTK_PRECISION_DOUBLE
00328
00362 void dormqr_(char *side,
00363              char *trans,
00364              int *m,
00365              int *n,
00366              int *k,
00367              Real *a,
00368              int *lda,
00369              Real *tau,
00370              Real *c,
00371              int *ldc,
00372              Real *work,
00373              int *lwork,
00374              int *info);
```

```
00375 #else
00376
00410 void sormqr_(char *side,
00411             char *trans,
00412             int *m,
00413             int *n,
00414             int *k,
00415             Real *a,
00416             int *lda,
00417             Real *tau,
00418             Real *c,
00419             int *ldc,
00420             Real *work,
00421             int *lwork,
00422             int *info);
00423 #endif
00424 }
00425 }
00426
00427 int mtk::LAPACKAdapter::SolveDenseSystem(
      mtk::DenseMatrix &mm,
00428                                         mtk::Real *rhs) {
00429
00430
00431   #if MTK_DEBUG_LEVEL > 0
00432   mtk::Tools::Prevent(rhs == nullptr, __FILE__, __LINE__, __func__);
00433   #endif
00434
00435   int *ipiv{};               // Array for pivoting information.
00436   int nrhs{1};               // Number of right-hand sides.
00437   int info{};                // Status of the solution.
00438   int mm_rank{mm.num_rows()}; // Rank of the matrix.
00439
00440   try {
00441     ipiv = new int[mm_rank];
00442   } catch (std::bad_alloc &memory_allocation_exception) {
00443     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00444       std::endl;
00445     std::cerr << memory_allocation_exception.what() << std::endl;
00446   }
00447   memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00448
00449   int ldbb = mm_rank;
00450   int mm_ld = mm_rank;
00451
00452   #ifdef MTK_PRECISION_DOUBLE
00453   dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00454   #else
00455   fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, rhs, &ldbb, &info);
00456   #endif
00457
00458   delete [] ipiv;
00459
00460   return info;
00461 }
00462
00463 int mtk::LAPACKAdapter::SolveDenseSystem(
      mtk::DenseMatrix &mm,
00464                                         mtk::DenseMatrix &bb) {
00465
00466   int nrhs{bb.num_rows()};  // Number of right-hand sides.
00467
00468   #if MTK_DEBUG_LEVEL > 0
00469   mtk::Tools::Prevent(nrhs <= 0, __FILE__, __LINE__, __func__);
00470   #endif
00471
00472   int *ipiv{};               // Array for pivoting information.
00473   int info{};                // Status of the solution.
00474   int mm_rank{mm.num_rows()}; // Rank of the matrix.
00475
00476   try {
00477     ipiv = new int[mm_rank];
00478   } catch (std::bad_alloc &memory_allocation_exception) {
00479     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00480       std::endl;
00481     std::cerr << memory_allocation_exception.what() << std::endl;
00482   }
00483   memset(ipiv, 0, sizeof(ipiv[0])*mm_rank);
00484
00485   int ldbb = mm_rank;
00486   int mm_ld = mm_rank;
```

```
00487
00488   #ifdef MTK_PRECISION_DOUBLE
00489   dgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00490   #else
00491   fgesv_(&mm_rank, &nrhs, mm.data(), &mm_ld, ipiv, bb.data(), &ldbb, &info);
00492   #endif
00493
00494   delete [] ipiv;
00495
00496   // After output, the data in the matrix will be column-major ordered.
00497
00498   bb.SetOrdering(mtk::COL_MAJOR);
00499
00500   #if MTK_DEBUG_LEVEL > 0
00501   std::cout << "bb_col_maj_ord =" << std::endl;
00502   std::cout << bb << std::endl;
00503   #endif
00504
00505   bb.OrderRowMajor();
00506
00507   #if MTK_DEBUG_LEVEL > 0
00508   std::cout << "bb_row_maj_ord =" << std::endl;
00509   std::cout << bb << std::endl;
00510   #endif
00511
00512   return info;
00513 }
00514
00515 mtk::DenseMatrix mtk::LAPACKAdapter::QRFactorDenseMatrix
      (mtk::DenseMatrix &aa) {
00516
00517   mtk::Real *work{}; // Working array.
00518   mtk::Real *tau{};  // Array for the Householder scalars.
00519
00520   // Prepare to factorize: allocate and inquire for the value of lwork.
00521   try {
00522     work = new mtk::Real[1];
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(work, mtk::kZero, sizeof(aa.data()[0])*1);
00529
00530   int lwork{-1};
00531   int info{};
00532
00533   int aa_num_cols = aa.num_cols();
00534   int aaT_num_rows = aa.num_cols();
00535   int aaT_num_cols = aa.num_rows();
00536
00537   #if MTK_DEBUG_LEVEL > 0
00538   std::cout << "Input matrix BEFORE QR factorization:" << std::endl;
00539   std::cout << aa << std::endl;
00540   #endif
00541
00542   #ifdef MTK_PRECISION_DOUBLE
00543   dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00544           tau,
00545           work, &lwork, &info);
00546   #else
00547   fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00548           tau,
00549           work, &lwork, &info);
00550   #endif
00551
00552   #if MTK_DEBUG_LEVEL > 0
00553   if (info == 0) {
00554     lwork = (int) work[0];
00555   } else {
00556     std::cerr << "Could not get value for lwork on line " << __LINE__ - 5 <<
00557       std::endl;
00558     std::cerr << "Exiting..." << std::endl;
00559   }
00560   #endif
00561
00562   #if MTK_DEBUG_LEVEL>0
00563   std::cout << "lwork = " << std::endl << std::setw(12) << lwork << std::endl
00564     << std::endl;
00565   #endif
00566
```

```
00567    delete [] work;
00568    work = nullptr;
00569
00570    // Once we know lwork, we can actually invoke the factorization:
00571    try {
00572      work = new mtk::Real [lwork];
00573    } catch (std::bad_alloc &memory_allocation_exception) {
00574      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00575        std::endl;
00576      std::cerr << memory_allocation_exception.what() << std::endl;
00577    }
00578    memset(work, mtk::kZero, sizeof(work[0])*lwork);
00579
00580    int ltau = std::min(aaT_num_rows,aaT_num_cols);
00581
00582    try {
00583      tau = new mtk::Real [ltau];
00584    } catch (std::bad_alloc &memory_allocation_exception) {
00585      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00586        std::endl;
00587      std::cerr << memory_allocation_exception.what() << std::endl;
00588    }
00589    memset(tau, mtk::kZero, sizeof(0.0)*ltau);
00590
00591    #ifdef MTK_PRECISION_DOUBLE
00592    dgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00593            tau, work, &lwork, &info);
00594    #else
00595    fgeqrf_(&aaT_num_rows, &aaT_num_cols, aa.data(), &aaT_num_rows,
00596            tau, work, &lwork, &info);
00597    #endif
00598
00599    if (!info) {
00600      #if MTK_DEBUG_LEVEL > 0
00601      std::cout << "QR factorization completed!" << std::endl << std::endl;
00602      #endif
00603    } else {
00604      std::cerr << "Error solving system! info = " << info << std::endl;
00605      std::cerr << "Exiting..." << std::endl;
00606    }
00607
00608    #if MTK_DEBUG_LEVEL > 0
00609    std::cout << "Input matrix AFTER QR factorization:" << std::endl;
00610    std::cout << aa << std::endl;
00611    #endif
00612
00613    // We now generate the real matrix Q with orthonormal columns. This has to
00614    // be done separately since the actual output of dgeqrf_ (AA_) represents
00615    // the orthogonal matrix Q as a product of min(aa_num_rows,aa_num_cols)
00616    // elementary Householder reflectors. Notice that we must re-inquire the new
00617    // value for lwork that is used.
00618
00619    bool padded{false};
00620
00621    bool transpose{false};
00622
00623    mtk::DenseMatrix QQ_(aa.num_cols(),padded,transpose);
00624
00625    #if MTK_DEBUG_LEVEL > 0
00626    std::cout << "Initialized QQ_T: " << std::endl;
00627    std::cout << QQ_ << std::endl;
00628    #endif
00629
00630    // Assemble the QQ_ matrix:
00631    lwork = -1;
00632
00633    delete[] work;
00634    work = nullptr;
00635
00636    try {
00637      work = new mtk::Real[1];
00638    } catch (std::bad_alloc &memory_allocation_exception) {
00639      std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00640        std::endl;
00641      std::cerr << memory_allocation_exception.what() <<
00642        std::endl;
00643    }
00644    memset(work, mtk::kZero, sizeof(work[0])*1);
00645
00646    char side_{'L'};
00647    char trans_{'N'};
```

```
00648
00649   int aux = QQ_.num_rows();
00650
00651   #ifdef MTK_PRECISION_DOUBLE
00652   dormqr_(&side_, &trans_,
00653           &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00654           QQ_.data(), &aux, work, &lwork, &info);
00655   #else
00656   formqr_(&side_, &trans_,
00657           &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00658           QQ_.data(), &aux, work, &lwork, &info);
00659   #endif
00660
00661   #if MTK_DEBUG_LEVEL > 0
00662   if (info == 0) {
00663     lwork = (int) work[0];
00664   } else {
00665     std::cerr << "Could not get lwork on line " << __LINE__ - 5 << std::endl;
00666     std::cerr << "Exiting..." << std::endl;
00667   }
00668   #endif
00669
00670   #if MTK_DEBUG_LEVEL > 0
00671   std::cout << "lwork = " << std::endl << std::setw(12) << lwork <<
00672     std::endl << std::endl;
00673   #endif
00674
00675   delete[] work;
00676   work = nullptr;
00677
00678   try {
00679     work = new mtk::Real[lwork];
00680   } catch (std::bad_alloc &memory_allocation_exception) {
00681     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 <<
00682       std::endl;
00683     std::cerr << memory_allocation_exception.what() << std::endl;
00684   }
00685   memset(work, mtk::kZero, sizeof(work[0])*lwork);
00686
00687   #ifdef MTK_PRECISION_DOUBLE
00688   dormqr_(&side_, &trans_,
00689           &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00690           QQ_.data(), &aux, work, &lwork, &info);
00691   #else
00692   formqr_(&side_, &trans_,
00693           &aa_num_cols, &aa_num_cols, &ltau, aa.data(), &aaT_num_rows, tau,
00694           QQ_.data(), &aux, work, &lwork, &info);
00695   #endif
00696
00697   if (!info) {
00698     #if MTK_DEBUG_LEVEL>0
00699     std::cout << "Q matrix successfully assembled!" << std::endl << std::endl;
00700     #endif
00701   } else {
00702     std::cerr << "Something went wrong solving system! info = " << info <<
00703       std::endl;
00704     std::cerr << "Exiting..." << std::endl;
00705   }
00706
00707   delete[] work;
00708   work = nullptr;
00709
00710   delete[] tau;
00711   tau = nullptr;
00712
00713   return QQ_;
00714 }
00715
00716 int mtk::LAPACKAdapter::SolveRectangularDenseSystem(const
      mtk::DenseMatrix &aa,
00717                                                     mtk::Real *ob_,
00718                                                     int ob_ld_) {
00719
00720   // We first invoke the solver to query for the value of lwork. For this,
00721   // we must at least allocate enough space to allow access to WORK(1), or
00722   // // work[0]:
00723
00724   // If LWORK = -1, then a workspace query is assumed; the routine only
00725   // calculates the optimal size of the WORK array, returns this value as
00726   // the first entry of the WORK array, and no error message related to
00727   // LWORK is issued by XERBLA.
```

```
00728
00729   mtk::Real *work{}; // Work array.
00730
00731   try {
00732     work = new mtk::Real[1];
00733   } catch (std::bad_alloc &memory_allocation_exception) {
00734     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00735     std::cerr << memory_allocation_exception.what() << std::endl;
00736   }
00737   memset(work, mtk::kZero, sizeof(work[0])*1);
00738
00739   char trans_{'N'};
00740   int nrhs_{1};
00741   int info{0};
00742   int lwork{-1};
00743
00744   int AA_num_rows_  = aa.num_cols();
00745   int AA_num_cols_  = aa.num_rows();
00746   int AA_ld_ = std::max(1,aa.num_cols());
00747
00748   #ifdef MTK_PRECISION_DOUBLE
00749   dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00750          ob_, &ob_ld_,
00751          work, &lwork, &info);
00752   #else
00753   sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00754          ob_, &ob_ld_,
00755          work, &lwork, &info);
00756   #endif
00757
00758   if (info == 0) {
00759     lwork = (int) work[0];
00760   } else {
00761     std::cerr << "Could not get value for lwork on line " << __LINE__ - 2 <<
00762       std::endl;
00763     std::cerr << "Exiting..." << std::endl;
00764     return info;
00765   }
00766
00767   #if MTK_DEBUG_LEVEL > 0
00768   std::cout << "lwork = " << std::endl << std::setw(12)<< lwork <<
00769     std::endl << std::endl;
00770   #endif
00771
00772   // We then use lwork's new value to create the work array:
00773   delete[] work;
00774   work = nullptr;
00775
00776   try {
00777     work = new mtk::Real[lwork];
00778   } catch (std::bad_alloc &memory_allocation_exception) {
00779     std::cerr << "Memory allocation exception on line " << __LINE__ - 3 << std::endl;
00780     std::cerr << memory_allocation_exception.what() << std::endl;
00781   }
00782   memset(work, 0.0, sizeof(work[0])*lwork);
00783
00784   // We now invoke the solver again:
00785   #ifdef MTK_PRECISION_DOUBLE
00786   dgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00787          ob_, &ob_ld_,
00788          work, &lwork, &info);
00789   #else
00790   sgels_(&trans_, &AA_num_rows_, &AA_num_cols_, &nrhs_, aa.data(), &AA_ld_,
00791          ob_, &ob_ld_,
00792          work, &lwork, &info);
00793   #endif
00794
00795   delete [] work;
00796   work = nullptr;
00797
00798   return info;
00799 }
```

17.49   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:
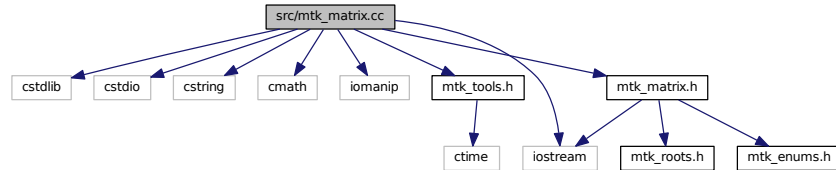
17.49.1 Detailed Description

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.

17.50 mtk_matrix.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to the source code should be reported to:
00018
00019 esanchez at mail dot sdsu dot edu
00020
00021 A copy of the modified files should be reported once modifications are
00022 completed. Documentation related to said modifications should be included.
00023
00024 2. Redistributions of source code must be done through direct
00025 downloads from the project's GitHub page:
00026
00027 http://www.csrc.sdsu.edu/mtk
00028
00029 3. Redistributions of source code must retain the above copyright notice, this
00030 list of conditions and the following disclaimer.
00031
00032 4. Redistributions in binary form must reproduce the above copyright notice,
00033 this list of conditions and the following disclaimer in the documentation and/or
00034 other materials provided with the distribution.
00035
00036 5. Usage of the binary form on proprietary applications shall require explicit
00037 prior written permission from the the copyright holders.
```

```
00038
00039 6. Neither the name of the copyright holder nor the names of its contributors
00040 may be used to endorse or promote products derived from this software without
00041 specific prior written permission.
00042
00043 The copyright holders provide no reassurances that the source code provided does
00044 not infringe any patent, copyright, or any other intellectual property rights of
00045 third parties. The copyright holders disclaim any liability to any recipient for
00046 claims brought against recipient by any third party for infringement of that
00047 parties intellectual property rights.
00048
00049 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00050 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00051 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00052 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00053 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00054 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00055 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00056 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00057 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00058 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00059 */
00060
00061 #include <cstdlib>
00062 #include <cstdio>
00063 #include <cstring>
00064 #include <cmath>
00065
00066 #include <iomanip>
00067 #include <iostream>
00068
00069 #include "mtk_tools.h"
00070 #include "mtk_matrix.h"
00071
00072 mtk::Matrix::Matrix():
00073   storage_(mtk::DENSE),
00074   ordering_(mtk::ROW_MAJOR),
00075   num_rows_(),
00076   num_cols_(),
00077   num_values_(),
00078   ld_(),
00079   num_zero_(),
00080   num_non_zero_(),
00081   num_null_(),
00082   num_non_null_(),
00083   kl_(),
00084   ku_(),
00085   bandwidth_(),
00086   abs_density_(),
00087   rel_density_(),
00088   abs_sparsity_(),
00089   rel_sparsity_() {}
00090
00091 mtk::Matrix::Matrix(const Matrix &in):
00092   storage_(in.storage_),
00093   ordering_(in.ordering_),
00094   num_rows_(in.num_rows_),
00095   num_cols_(in.num_cols_),
00096   num_values_(in.num_values_),
00097   ld_(in.ld_),
00098   num_zero_(in.num_zero_),
00099   num_non_zero_(in.num_non_zero_),
00100   num_null_(in.num_null_),
00101   num_non_null_(in.num_non_null_),
00102   kl_(in.kl_),
00103   ku_(in.ku_),
00104   bandwidth_(in.bandwidth_),
00105   abs_density_(in.abs_density_),
00106   rel_density_(in.rel_density_),
00107   abs_sparsity_(in.abs_sparsity_),
00108   rel_sparsity_(in.rel_sparsity_) {}
00109
00110 mtk::Matrix::~Matrix() {}
00111
00112 mtk::MatrixStorage mtk::Matrix::storage() const {
00113
00114   return storage_;
00115 }
00116
00117 mtk::MatrixOrdering mtk::Matrix::ordering() const {
00118
```

```
00119    return ordering_;
00120 }
00121
00122 int mtk::Matrix::num_rows() const {
00123
00124    return num_rows_;
00125 }
00126
00127 int mtk::Matrix::num_cols() const {
00128
00129    return num_cols_;
00130 }
00131
00132 int mtk::Matrix::num_values() const {
00133
00134    return num_values_;
00135 }
00136
00137 int mtk::Matrix::ld() const {
00138
00139    return ld_;
00140 }
00141
00142 int mtk::Matrix::num_zero() const {
00143
00144    return num_zero_;
00145 }
00146
00147 int mtk::Matrix::num_non_zero() const {
00148
00149    return num_non_zero_;
00150 }
00151
00152 int mtk::Matrix::num_null() const {
00153
00154    return num_null_;
00155 }
00156
00157 int mtk::Matrix::num_non_null() const {
00158
00159    return num_non_null_;
00160 }
00161
00162 int mtk::Matrix::kl() const {
00163
00164    return kl_;
00165 }
00166
00167 int mtk::Matrix::ku() const {
00168
00169    return ku_;
00170 }
00171
00172 int mtk::Matrix::bandwidth() const {
00173
00174    return bandwidth_;
00175 }
00176
00177 mtk::Real mtk::Matrix::rel_density() const {
00178
00179    return rel_density_;
00180 }
00181
00182 mtk::Real mtk::Matrix::abs_sparsity() const {
00183
00184    return abs_sparsity_;
00185 }
00186
00187 mtk::Real mtk::Matrix::rel_sparsity() const {
00188
00189    return rel_sparsity_;
00190 }
00191
00192 void mtk::Matrix::set_storage(const mtk::MatrixStorage &ss) {
00193
00194    #if MTK_DEBUG_LEVEL > 0
00195    mtk::Tools::Prevent(!(ss == mtk::DENSE ||
00196                          ss == mtk::BANDED ||
00197                          ss == mtk::CRS),
00198                        __FILE__, __LINE__, __func__);
00199    #endif
```

```
00200
00201   storage_ = ss;
00202 }
00203
00204 void mtk::Matrix::set_ordering(const
      mtk::MatrixOrdering &oo) {
00205
00206   #if MTK_DEBUG_LEVEL > 0
00207   mtk::Tools::Prevent(!(oo == mtk::ROW_MAJOR || oo ==
      mtk::COL_MAJOR),
00208                       __FILE__, __LINE__, __func__);
00209   #endif
00210
00211   ordering_ = oo;
00212
00213   ld_ = (ordering_ == mtk::ROW_MAJOR)?
00214     std::max(1,num_cols_): std::max(1,num_rows_);
00215 }
00216
00217 void mtk::Matrix::set_num_rows(int in) {
00218
00219   #if MTK_DEBUG_LEVEL > 0
00220   mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00221   #endif
00222
00223   num_rows_ = in;
00224   num_values_ = num_rows_*num_cols_;
00225   ld_ = (ordering_ == mtk::ROW_MAJOR)?
00226     std::max(1,num_cols_): std::max(1,num_rows_);
00227 }
00228
00229 void mtk::Matrix::set_num_cols(int in) {
00230
00231   #if MTK_DEBUG_LEVEL > 0
00232   mtk::Tools::Prevent(in < 1, __FILE__, __LINE__, __func__);
00233   #endif
00234
00235   num_cols_ = in;
00236   num_values_ = num_rows_*num_cols_;
00237   ld_ = (ordering_ == mtk::ROW_MAJOR)?
00238     std::max(1,num_cols_): std::max(1,num_rows_);
00239 }
00240
00241 void mtk::Matrix::set_num_zero(int in) {
00242
00243   #if MTK_DEBUG_LEVEL > 0
00244   mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00245   #endif
00246
00247   num_zero_ = in;
00248   num_non_zero_ = num_values_ - num_zero_;
00249
00251   rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00252   rel_sparsity_ = 1.0 - rel_density_;
00253 }
00254
00255 void mtk::Matrix::set_num_null(int in) {
00256
00257   #if MTK_DEBUG_LEVEL > 0
00258   mtk::Tools::Prevent(in < 0, __FILE__, __LINE__, __func__);
00259   #endif
00260
00261   num_null_ = in;
00262   num_non_null_ = num_values_ - num_null_;
00263
00265   abs_density_ = (mtk::Real) num_non_null_/num_values_;
00266   abs_sparsity_ = 1.0 - abs_density_;
00267 }
00268
00269 void mtk::Matrix::IncreaseNumZero() {
00270
00272
00273   num_zero_++;
00274   num_non_zero_ = num_values_ - num_zero_;
00275   rel_density_ = (mtk::Real) num_non_zero_/num_values_;
00276   rel_sparsity_ = 1.0 - rel_density_;
00277 }
00278
00279 void mtk::Matrix::IncreaseNumNull() {
00280
00282
```

```
00283   num_null_++;
00284   num_non_null_ = num_values_ - num_null_;
00285   abs_density_ = (mtk::Real) num_non_null_/num_values_;
00286   abs_sparsity_ = 1.0 - abs_density_;
00287 }
```

17.51   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:
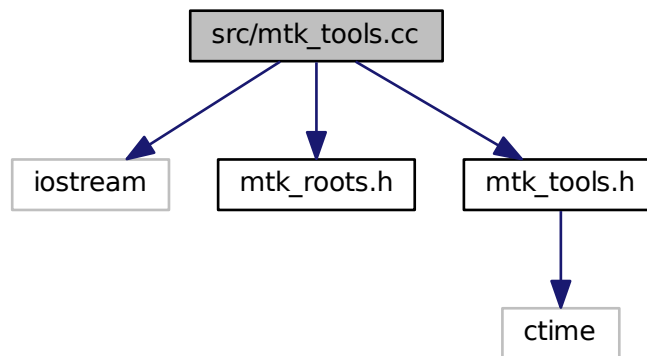
17.51.1   Detailed Description

Basic tools to ensure execution correctness.

Author

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

Definition in file mtk_tools.cc.

17.52   mtk_tools.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
```

```
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057
00058 #include "mtk_roots.h"
00059 #include "mtk_tools.h"
00060
00061 void mtk::Tools::Prevent(const bool condition,
00062                          const char *fname,
00063                          int lineno,
00064                          const char *fxname) {
00065
00066
00067   #if MTK_DEBUG_LEVEL > 0
00068   if (lineno < 1) {
00069     std::cerr << __FILE__ << ": " << "Incorrect parameter at line " <<
00070     __LINE__ - 2 << " (" << __func__ << ")" << std::endl;
00071     exit(EXIT_FAILURE);
00072   }
00073   #endif
00074
00075   if (condition) {
00076     std::cerr << fname << ": " << "Incorrect parameter at line " <<
00077     lineno << " (" << fxname << ")" << std::endl;
00078     exit(EXIT_FAILURE);
00079   }
00080 }
00081
00082
00083 int mtk::Tools::test_number_; // Used to control the correctness of the test.
00084
00085 clock_t mtk::Tools::begin_time_;  // Used to time tests.
00086
00087 void mtk::Tools::BeginTestNo(const int &nn) {
00088
00089   #if MTK_DEBUG_LEVEL > 0
00090   mtk::Tools::Prevent(nn <= 0, __FILE__, __LINE__, __func__);
00091   #endif
00092
00093   test_number_ = nn;
00094
00095   std::cout << "Test " << nn << "..." << std::endl << std::endl;
00096   begin_time_ = clock();
00097 }
```

```
00100
00101 void mtk::Tools::EndTestNo(const int &nn) {
00102
00103   #if MTK_DEBUG_LEVEL > 0
00104   mtk::Tools::Prevent(test_number_ != nn, __FILE__, __LINE__, __func__);
00105   #endif
00106
00107   auto duration = mtk::Real(clock() - begin_time_)/CLOCKS_PER_SEC;
00108   std::cout << "Test " << test_number_ << " complete! ";
00109   std::cout << "Elapsed: " << duration << " seconds." << std::endl;
00110 }
```

17.53 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"
```
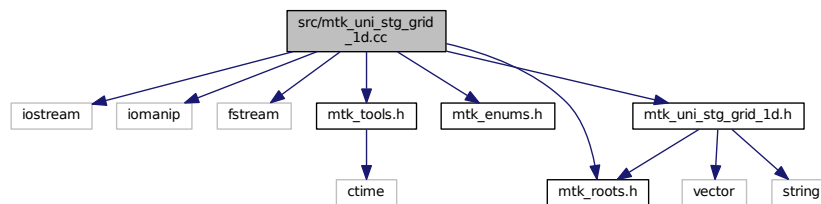
Include dependency graph for mtk_uni_stg_grid_1d.cc:

Namespaces

- mtk

    Mimetic Methods Toolkit namespace.

Functions

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

17.53.1 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.

17.54   mtk_uni_stg_grid_1d.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #include <iostream>
00057 #include <iomanip>
00058 #include <fstream>
00059
00060 #include "mtk_roots.h"
00061 #include "mtk_enums.h"
00062 #include "mtk_tools.h"
00063
00064 #include "mtk_uni_stg_grid_1d.h"
00065
00066 namespace mtk {
00067
00068 std::ostream& operator <<(std::ostream &stream, mtk::UniStgGrid1D &in) {
00069
00070   stream << '[' << in.west_bndy_x_ << ':' << in.num_cells_x_ << ':' <<
00071   in.east_bndy_x_ << "] = " << std::endl << std::endl;
00072
00074
00075   stream << "x:";
00076   for (unsigned int ii = 0; ii < in.discrete_domain_x_.size(); ++ii) {
00077     stream << std::setw(10) << in.discrete_domain_x_[ii];
00078   }
00079   stream << std::endl;
00080
00081   if (in.nature_ == mtk::SCALAR) {
00082     stream << "u:";
00083   }
00084   else {
00085     stream << "v:";
00086   }
00087   for (unsigned int ii = 0; ii < in.discrete_field_u_.size(); ++ii) {
```

```
00088       stream << std::setw(10) << in.discrete_field_u_[ii];
00089    }
00090
00091    stream << std::endl;
00092
00093    return stream;
00094 }
00095 }
00096
00097 mtk::UniStgGrid1D::UniStgGrid1D():
00098       nature_(),
00099       discrete_domain_x_(),
00100       discrete_field_u_(),
00101       west_bndy_x_(),
00102       east_bndy_x_(),
00103       num_cells_x_(),
00104       delta_x_() {}
00105
00106 mtk::UniStgGrid1D::UniStgGrid1D(const
     UniStgGrid1D &grid):
00107       nature_(grid.nature_),
00108       west_bndy_x_(grid.west_bndy_x_),
00109       east_bndy_x_(grid.east_bndy_x_),
00110       num_cells_x_(grid.num_cells_x_),
00111       delta_x_(grid.delta_x_) {
00112
00113       std::copy(grid.discrete_domain_x_.begin(),
00114                 grid.discrete_domain_x_.begin() + grid.
     discrete_domain_x_.size(),
00115                 discrete_domain_x_.begin());
00116
00117       std::copy(grid.discrete_field_u_.begin(),
00118                 grid.discrete_field_u_.begin() + grid.
     discrete_field_u_.size(),
00119                 discrete_field_u_.begin());
00120 }
00121
00122 mtk::UniStgGrid1D::UniStgGrid1D(const Real &west_bndy_x,
00123                                const Real &east_bndy_x,
00124                                const int &num_cells_x,
00125                                const mtk::FieldNature &nature) {
00126
00127    #if MTK_DEBUG_LEVEL > 0
00128    mtk::Tools::Prevent(west_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00129    mtk::Tools::Prevent(east_bndy_x < mtk::kZero, __FILE__, __LINE__, __func__);
00130    mtk::Tools::Prevent(east_bndy_x <= west_bndy_x, __FILE__, __LINE__, __func__);
00131    mtk::Tools::Prevent(num_cells_x < 0, __FILE__, __LINE__, __func__);
00132    #endif
00133
00134    nature_ = nature;
00135    west_bndy_x_ = west_bndy_x;
00136    east_bndy_x_ = east_bndy_x;
00137    num_cells_x_ = num_cells_x;
00138
00139    delta_x_ = (east_bndy_x - west_bndy_x)/((mtk::Real) num_cells_x);
00140 }
00141
00142 mtk::UniStgGrid1D::~UniStgGrid1D() {}
00143
00144 mtk::Real mtk::UniStgGrid1D::delta_x() const {
00145
00146    return delta_x_;
00147 }
00148
00149 mtk::Real *mtk::UniStgGrid1D::discrete_domain_x() {
00150
00151    return discrete_domain_x_.data();
00152 }
00153
00154 mtk::Real *mtk::UniStgGrid1D::discrete_field_u() {
00155
00156    return discrete_field_u_.data();
00157 }
00158
00159 int mtk::UniStgGrid1D::num_cells_x() const {
00160
00161    return num_cells_x_;
00162 }
00163
00164 void mtk::UniStgGrid1D::BindScalarField(
00165       mtk::Real (*ScalarField)(mtk::Real xx)) {
```

```
00166
00167    #if MTK_DEBUG_LEVEL > 0
00168    mtk::Tools::Prevent(nature_ == mtk::VECTOR, __FILE__, __LINE__, __func__);
00169    #endif
00170
00172
00173    discrete_domain_x_.reserve(num_cells_x_ + 2);
00174
00175    discrete_domain_x_.push_back(west_bndy_x_);
00176    #ifdef MTK_PRECISION_DOUBLE
00177    auto first_center = west_bndy_x_ + delta_x_/2.0;
00178    #else
00179    auto first_center = west_bndy_x_ + delta_x_/2.0f;
00180    #endif
00181    discrete_domain_x_.push_back(first_center);
00182    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00183      discrete_domain_x_.push_back(first_center + ii*delta_x_);
00184    }
00185    discrete_domain_x_.push_back(east_bndy_x_);
00186
00188
00189    discrete_field_u_.reserve(num_cells_x_ + 2);
00190
00191    discrete_field_u_.push_back(ScalarField(west_bndy_x_));
00192
00193    discrete_field_u_.push_back(ScalarField(first_center));
00194    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00195      discrete_field_u_.push_back(ScalarField(first_center + ii*delta_x_));
00196    }
00197    discrete_field_u_.push_back(ScalarField(east_bndy_x_));
00198 }
00199
00200 void mtk::UniStgGrid1D::BindVectorField(
00201      mtk::Real (*VectorField)(mtk::Real xx)) {
00202
00203    #if MTK_DEBUG_LEVEL > 0
00204    mtk::Tools::Prevent(nature_ == mtk::SCALAR, __FILE__, __LINE__, __func__);
00205    #endif
00206
00208
00209    discrete_domain_x_.reserve(num_cells_x_ + 1);
00210
00211    discrete_domain_x_.push_back(west_bndy_x_);
00212    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00213      discrete_domain_x_.push_back(west_bndy_x_ + ii*delta_x_);
00214    }
00215    discrete_domain_x_.push_back(east_bndy_x_);
00216
00218
00219    discrete_field_u_.reserve(num_cells_x_ + 1);
00220
00221    discrete_field_u_.push_back(VectorField(west_bndy_x_));
00222    for (auto ii = 1; ii < num_cells_x_; ++ii) {
00223      discrete_field_u_.push_back(VectorField(west_bndy_x_ + ii*delta_x_));
00224    }
00225    discrete_field_u_.push_back(VectorField(east_bndy_x_));
00226 }
00227
00228 bool mtk::UniStgGrid1D::WriteToFile(std::string filename,
00229                                    std::string space_name,
00230                                    std::string field_name) {
00231
00232    std::ofstream output_dat_file;  // Output file.
00233
00234    output_dat_file.open(filename);
00235
00236    if (!output_dat_file.is_open()) {
00237      return false;
00238    }
00239
00240    output_dat_file << "# " << space_name << ' ' << field_name << std::endl;
00241    for (unsigned int ii = 0; ii < discrete_domain_x_.size(); ++ii) {
00242      output_dat_file << discrete_domain_x_[ii] << ' ' << discrete_field_u_[ii] <<
00243        std::endl;
00244    }
00245
00246    output_dat_file.close();
00247
00248    return true;
00249 }
```

17.55 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:
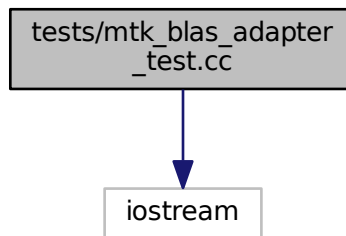
Functions

- int main ()

17.55.1 Detailed Description

Author

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

Definition in file mtk_blas_adapter_test.cc.

17.55.2 Function Documentation

17.55.2.1 int main ( )

Definition at line 107 of file mtk_blas_adapter_test.cc.

17.56 mtk_blas_adapter_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed. Documentation related to said modifications should be included.
```

```
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void Test1() {
00061
00062   mtk::Tools::BeginTestNo(1);
00063
00064   int rr = 2;
00065   int cc = 3;
00066
00067   mtk::DenseMatrix aa(rr,cc);
00068
00069   aa.SetValue(0,0,1.0);
00070   aa.SetValue(0,1,2.0);
00071   aa.SetValue(0,2,3.0);
00072   aa.SetValue(1,0,4.0);
00073   aa.SetValue(1,1,5.0);
00074   aa.SetValue(1,2,6.0);
00075
00076   std::cout << aa << std::endl;
00077
00078   mtk::DenseMatrix bb(cc,rr);
00079
00080   bb.SetValue(0,0,7.0);
00081   bb.SetValue(0,1,8.0);
00082   bb.SetValue(1,0,9.0);
00083   bb.SetValue(1,1,10.0);
00084   bb.SetValue(2,0,11.0);
00085   bb.SetValue(2,1,12.0);
00086
00087   std::cout << bb << std::endl;
00088
00089   mtk::DenseMatrix pp = mtk::BLASAdapter::RealDenseMM(aa,bb);
00090
00091   std::cout << pp << std::endl;
00092
00093   mtk::Tools::EndTestNo(1);
00094 }
00095
00096 int main () {
00097
00098   std::cout << "Testing mtk::BLASAdapter class." << std::endl;
```

```
00099
00100    Test1();
00101 }
00102
00103 #else
00104 #include <iostream>
00105 using std::cout;
00106 using std::endl;
00107 int main () {
00108    cout << "This code HAS to be compiled with support for C++11." << endl;
00109    cout << "Exiting..." << endl;
00110 }
00111 #endif
```

17.57 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:
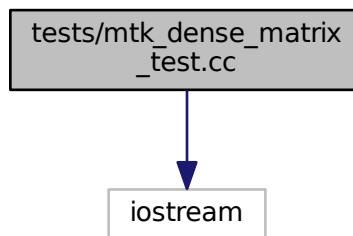
Functions

- int main ()

17.57.1 Detailed Description

Author

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

Definition in file mtk_dense_matrix_test.cc.

17.57.2 Function Documentation

17.57.2.1 int main ( )

Definition at line 285 of file mtk_dense_matrix_test.cc.

17.58   mtk_dense_matrix_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void Test1() {
00062
00063   mtk::Tools::BeginTestNo(1);
00064
00065   mtk::DenseMatrix m1;
00066
00067   std::cout << m1 << std::endl;
00068
00069   mtk::Tools::EndTestNo(1);
00070 }
00071
00072 void Test2() {
00073
00074   mtk::Tools::BeginTestNo(2);
00075
00076   int rr = 4;
00077   int cc = 7;
00078
00079   mtk::DenseMatrix m2(rr,cc);
00080
00081   std::cout << m2 << std::endl;
00082
00083   mtk::Tools::EndTestNo(2);
00084 }
```

```
00085
00086 void Test3() {
00087
00088   mtk::Tools::BeginTestNo(3);
00089
00090   int rank = 5;
00091   bool padded = true;
00092   bool transpose = false;
00093
00094   mtk::DenseMatrix m3(rank,padded,transpose);
00095
00096   std::cout << m3 << std::endl;
00097
00098   mtk::Tools::EndTestNo(3);
00099 }
00100
00101 void Test4() {
00102
00103   mtk::Tools::BeginTestNo(4);
00104
00105   int rank = 5;
00106   bool padded = false;
00107   bool transpose = false;
00108
00109   mtk::DenseMatrix m4(rank,padded,transpose);
00110
00111   std::cout << m4 << std::endl;
00112
00113   mtk::Tools::EndTestNo(4);
00114 }
00115
00116 void Test5() {
00117
00118   mtk::Tools::BeginTestNo(5);
00119
00120   int rr = 4;
00121   int cc = 7;
00122
00123   mtk::DenseMatrix m5(rr,cc);
00124
00125   for (auto ii = 0; ii < rr; ++ii) {
00126     for (auto jj = 0; jj < cc; ++jj) {
00127       m5.SetValue(ii,jj,(mtk::Real) ii + jj);
00128     }
00129   }
00130
00131   std::cout << m5 << std::endl;
00132
00133   mtk::Real *vals = m5.data();
00134
00135   for (auto ii = 0; ii < rr; ++ii) {
00136     for (auto jj = 0; jj < cc; ++jj) {
00137       std::cout << " " << vals[ii*cc + jj];
00138     }
00139     std::cout << std::endl;
00140   }
00141   std::cout << std::endl;
00142
00143   for (auto ii = 0; ii < rr; ++ii) {
00144     for (auto jj = 0; jj < cc; ++jj) {
00145       std::cout << " " << m5.GetValue(ii,jj);
00146     }
00147     std::cout << std::endl;
00148   }
00149   std::cout << std::endl;
00150
00151   mtk::Tools::EndTestNo(5);
00152 }
00153
00154 void Test6() {
00155
00156   mtk::Tools::BeginTestNo(6);
00157
00158   bool transpose = false;
00159   int generator_length = 3;
00160   int progression_length = 4;
00161
00162   mtk::Real generator[] = {-0.5, 0.5, 1.5};
00163
00164   mtk::DenseMatrix m6(generator,generator_length,progression_length,transpose);
00165
```

```
00166   std::cout << m6 << std::endl;
00167
00168   transpose = true;
00169
00170   mtk::DenseMatrix m7(generator,generator_length,progression_length,transpose);
00171
00172   std::cout << m7 << std::endl;
00173
00174
00175   mtk::Tools::EndTestNo(6);
00176 }
00177
00178 void Test7() {
00179
00180   mtk::Tools::BeginTestNo(7);
00181
00182   bool padded = false;
00183   bool transpose = false;
00184   int lots_of_rows = 2;
00185   int lots_of_cols = 5;
00186   mtk::DenseMatrix m8(lots_of_rows,padded,transpose);
00187
00188   std::cout << m8 << std::endl;
00189
00190   mtk::DenseMatrix m9(lots_of_rows,lots_of_cols);
00191
00192   for (auto ii = 0; ii < lots_of_rows; ++ii) {
00193     for (auto jj = 0; jj < lots_of_cols; ++jj) {
00194       m9.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00195     }
00196   }
00197
00198   std::cout << m9 << std::endl;
00199
00200   mtk::DenseMatrix m10 = mtk::DenseMatrix::Kron(m8,m9);
00201
00202   std::cout << m10 << std::endl;
00203
00204   mtk::Tools::EndTestNo(7);
00205 }
00206
00207 void Test8() {
00208
00209   mtk::Tools::BeginTestNo(8);
00210
00211   int lots_of_rows = 4;
00212   int lots_of_cols = 3;
00213   mtk::DenseMatrix m11(lots_of_rows,lots_of_cols);
00214
00215   for (auto ii = 0; ii < lots_of_rows; ++ii) {
00216     for (auto jj = 0; jj < lots_of_cols; ++jj) {
00217       m11.SetValue(ii,jj,(mtk::Real) ii*lots_of_cols + jj + 1);
00218     }
00219   }
00220
00221   std::cout << m11 << std::endl;
00222
00223   m11.Transpose();
00224
00225   std::cout << m11 << std::endl;
00226
00227   mtk::DenseMatrix m12;
00228
00229   m12 = m11;
00230
00231   std::cout << m12 << std::endl;
00232
00233   mtk::Tools::EndTestNo(8);
00234 }
00235
00236 void Test9() {
00237
00238   mtk::Tools::BeginTestNo(9);
00239
00240   bool transpose = false;
00241   int gg_l = 3;
00242   int progression_length = 4;
00243   mtk::Real gg[] = {-0.5, 0.5, 1.5};
00244
00245   mtk::DenseMatrix m13(gg, gg_l ,progression_length, transpose);
00246
```

```
00247    std::cout << m13 << std::endl;
00248
00249    mtk::DenseMatrix m14;
00250
00251    m14 = m13;
00252
00253    std::cout << m14 << std::endl;
00254
00255    m13.Transpose();
00256
00257    std::cout << m13 << std::endl;
00258
00259    m14 = m13;
00260
00261    std::cout << m14 << std::endl;
00262
00263    mtk::Tools::EndTestNo(9);
00264 }
00265
00266 int main () {
00267
00268    std::cout << "Testing mtk::DenseMatrix class." << std::endl;
00269
00270    Test1();
00271    Test2();
00272    Test3();
00273    Test4();
00274    Test5();
00275    Test6();
00276    Test7();
00277    Test8();
00278    Test9();
00279 }
00280
00281 #else
00282 #include <iostream>
00283 using std::cout;
00284 using std::endl;
00285 int main () {
00286    cout << "This code HAS to be compiled with support for C++11." << endl;
00287    cout << "Exiting..." << endl;
00288 }
00289 #endif
```

17.59   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:
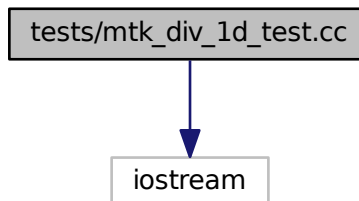
Functions

- int main ()

17.59.1 Detailed Description

Author

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

Definition in file mtk_div_1d_test.cc.

17.59.2 Function Documentation

17.59.2.1 int main ( )

Definition at line 248 of file mtk_div_1d_test.cc.

17.60 mtk_div_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
```

```
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void Test1() {
00061
00062   mtk::Tools::BeginTestNo(1);
00063
00064   mtk::Div1D div2;
00065
00066   bool info = div2.ConstructDiv1D();
00067
00068   if (!info) {
00069     std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00070   }
00071
00072   std::cout << div2 << std::endl;
00073
00074   mtk::Tools::EndTestNo(1);
00075 }
00076
00077 void Test2() {
00078
00079   mtk::Tools::BeginTestNo(2);
00080
00081   mtk::Div1D div4;
00082
00083   bool info = div4.ConstructDiv1D(4);
00084
00085   if (!info) {
00086     std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00087   }
00088
00089   std::cout << div4 << std::endl;
00090
00091   mtk::Tools::EndTestNo(2);
00092 }
00093
00094 void Test3() {
00095
00096   mtk::Tools::BeginTestNo(3);
00097
00098   mtk::Div1D div6;
00099
00100   bool info = div6.ConstructDiv1D(6);
00101
00102   if (!info) {
00103     std::cerr << "Mimetic div (6th order) could not be built." << std::endl;
00104   }
00105
00106   std::cout << div6 << std::endl;
00107
00108   mtk::Tools::EndTestNo(3);
00109 }
00110
00111 void Test4() {
00112
00113   mtk::Tools::BeginTestNo(4);
00114
00115   mtk::Div1D div8;
00116
00117   bool info = div8.ConstructDiv1D(8);
00118
00119   if (!info) {
00120     std::cerr << "Mimetic div (8th order) could not be built." << std::endl;
00121   }
00122
00123   std::cout << div8 << std::endl;
00124
00125   mtk::Tools::EndTestNo(4);
00126 }
00127
00128 void Test5() {
00129
00130   mtk::Tools::BeginTestNo(5);
00131
00132   mtk::Div1D div10;
00133
00134   bool info = div10.ConstructDiv1D(10);
```

```
00135
00136   if (!info) {
00137     std::cerr << "Mimetic div (10th order) could not be built." << std::endl;
00138   }
00139
00140   std::cout << div10 << std::endl;
00141
00142   mtk::Tools::EndTestNo(5);
00143 }
00144
00145 void Test6() {
00146
00147   mtk::Tools::BeginTestNo(6);
00148
00149   mtk::Div1D div12;
00150
00151   bool info = div12.ConstructDiv1D(12);
00152
00153   if (!info) {
00154     std::cerr << "Mimetic div (12th order) could not be built." << std::endl;
00155   }
00156
00157   std::cout << div12 << std::endl;
00158
00159   mtk::Tools::EndTestNo(6);
00160 }
00161
00162 void Test7() {
00163
00164   mtk::Tools::BeginTestNo(7);
00165
00166   mtk::Div1D div14;
00167
00168   bool info = div14.ConstructDiv1D(14);
00169
00170   if (!info) {
00171     std::cerr << "Mimetic div (14th order) could not be built." << std::endl;
00172   }
00173
00174   std::cout << div14 << std::endl;
00175
00176   mtk::Tools::EndTestNo(7);
00177 }
00178
00179 void Test8() {
00180
00181   mtk::Tools::BeginTestNo(8);
00182
00183   mtk::Div1D div2;
00184
00185   bool info = div2.ConstructDiv1D();
00186
00187   if (!info) {
00188     std::cerr << "Mimetic div (2nd order) could not be built." << std::endl;
00189   }
00190
00191   std::cout << div2 << std::endl;
00192
00193   mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00194
00195   std::cout << grid << std::endl;
00196
00197   mtk::DenseMatrix div2m(div2.ReturnAsDenseMatrix(grid));
00198
00199   std::cout << div2m << std::endl;
00200
00201   mtk::Tools::EndTestNo(8);
00202 }
00203
00204 void Test9() {
00205
00206   mtk::Tools::BeginTestNo(9);
00207
00208   mtk::Div1D div4;
00209
00210   bool info = div4.ConstructDiv1D(4);
00211
00212   if (!info) {
00213     std::cerr << "Mimetic div (4th order) could not be built." << std::endl;
00214   }
00215
```

```
00216   std::cout << div4 << std::endl;
00217
00218   mtk::UniStgGrid1D grid(0.0, 1.0, 11);
00219
00220   std::cout << grid << std::endl;
00221
00222   mtk::DenseMatrix div4m(div4.ReturnAsDenseMatrix(grid));
00223
00224   std::cout << div4m << std::endl;
00225
00226   mtk::Tools::EndTestNo(9);
00227 }
00228
00229 int main () {
00230
00231   std::cout << "Testing mtk::Div1D class." << std::endl;
00232
00233   Test1();
00234   Test2();
00235   Test3();
00236   Test4();
00237   Test5();
00238   Test6();
00239   Test7();
00240   Test8();
00241   Test9();
00242 }
00243
00244 #else
00245 #include <iostream>
00246 using std::cout;
00247 using std::endl;
00248 int main () {
00249   cout << "This code HAS to be compiled with support for C++11." << endl;
00250   cout << "Exiting..." << endl;
00251 }
00252 #endif
```

17.61   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:
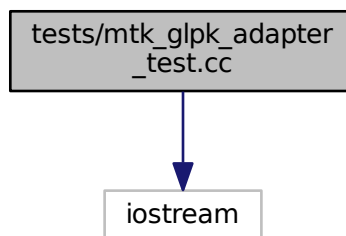
Functions

- int main ()

17.61.1 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.

17.61.2 Function Documentation

17.61.2.1 int main ( )

Definition at line 81 of file mtk_glpk_adapter_test.cc.

17.62 mtk_glpk_adapter_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
```

```
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065   mtk::Tools::BeginTestNo(1);
00066
00067   mtk::Tools::EndTestNo(1);
00068 }
00069
00070 int main () {
00071
00072   std::cout << "Testing mtk::GLPKAdapter class." << std::endl;
00073
00074   Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082   cout << "This code HAS to be compiled with support for C++11." << endl;
00083   cout << "Exiting..." << endl;
00084 }
00085 #endif
```

17.63 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:
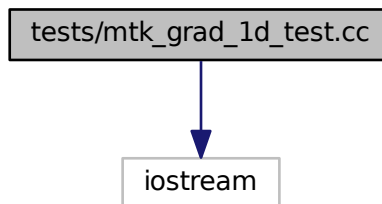
Functions

- int main ()

17.63.1 Detailed Description

Author

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

Definition in file mtk_grad_1d_test.cc.

17.63.2 Function Documentation

17.63.2.1 int main ( )

Definition at line 186 of file mtk_grad_1d_test.cc.

17.64 mtk_grad_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed. Documentation related to said modifications should be included.
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057
00058 #include "mtk.h"
00059
00060 void Test1() {
00061
00062   mtk::Tools::BeginTestNo(1);
00063
00064   mtk::Grad1D grad2;
00065
00066   bool info = grad2.ConstructGrad1D();
00067
00068   if (!info) {
00069     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
00070   }
00071
00072   std::cout << grad2 << std::endl;
00073
```

```
00074   mtk::Tools::EndTestNo(1);
00075 }
00076
00077 void Test2() {
00078
00079   mtk::Tools::BeginTestNo(2);
00080
00081   mtk::Grad1D grad4;
00082
00083   bool info = grad4.ConstructGrad1D(4);
00084
00085   if (!info) {
00086     std::cerr << "Mimetic grad (4th order) could not be built." << std::endl;
00087   }
00088
00089   std::cout << grad4 << std::endl;
00090
00091   mtk::Tools::EndTestNo(2);
00092 }
00093
00094 void Test3() {
00095
00096   mtk::Tools::BeginTestNo(3);
00097
00098   mtk::Grad1D grad6;
00099
00100   bool info = grad6.ConstructGrad1D(6);
00101
00102   if (!info) {
00103     std::cerr << "Mimetic grad (6th order) could not be built." << std::endl;
00104   }
00105
00106   std::cout << grad6 << std::endl;
00107
00108   mtk::Tools::EndTestNo(3);
00109 }
00110
00111 void Test4() {
00112
00113   mtk::Tools::BeginTestNo(4);
00114
00115   mtk::Grad1D grad8;
00116
00117   bool info = grad8.ConstructGrad1D(8);
00118
00119   if (!info) {
00120     std::cerr << "Mimetic grad (8th order) could not be built." << std::endl;
00121   }
00122
00123   std::cout << grad8 << std::endl;
00124
00125   mtk::Tools::EndTestNo(4);
00126 }
00127
00128 void Test5() {
00129
00130   mtk::Tools::BeginTestNo(5);
00131
00132   mtk::Grad1D grad10;
00133
00134   bool info = grad10.ConstructGrad1D(10);
00135
00136   if (!info) {
00137     std::cerr << "Mimetic grad (10th order) could not be built." << std::endl;
00138   }
00139
00140   std::cout << grad10 << std::endl;
00141
00142   mtk::Tools::EndTestNo(5);
00143 }
00144
00145 void Test6() {
00146
00147   mtk::Tools::BeginTestNo(6);
00148
00149   mtk::Grad1D grad2;
00150
00151   bool info = grad2.ConstructGrad1D();
00152
00153   if (!info) {
00154     std::cerr << "Mimetic grad (2nd order) could not be built." << std::endl;
```

```
00155  }
00156
00157    std::cout << grad2 << std::endl;
00158
00159    mtk::UniStgGrid1D grid(0.0, 1.0, 5);
00160
00161    std::cout << grid << std::endl;
00162
00163    mtk::DenseMatrix grad2m(grad2.ReturnAsDenseMatrix(grid));
00164
00165    std::cout << grad2m << std::endl;
00166
00167    mtk::Tools::EndTestNo(6);
00168  }
00169
00170  int main () {
00171
00172    std::cout << "Testing mtk::Grad1D class." << std::endl;
00173
00174    Test1();
00175    Test2();
00176    Test3();
00177    Test4();
00178    Test5();
00179    Test6();
00180  }
00181
00182  #else
00183  #include <iostream>
00184  using std::cout;
00185  using std::endl;
00186  int main () {
00187    cout << "This code HAS to be compiled with support for C++11." << endl;
00188    cout << "Exiting..." << endl;
00189  }
00190  #endif
```

17.65 tests/mtk_lap_1d_test.cc File Reference

```
#include <iostream>
```
Include dependency graph for mtk_lap_1d_test.cc:
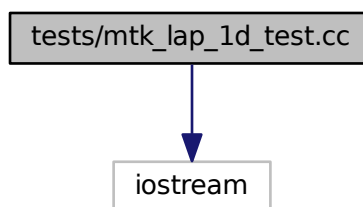
Functions

- int main ()

17.65.1 Function Documentation

17.65.1.1 int main ( )

Definition at line 156 of file mtk_lap_1d_test.cc.

17.66 mtk_lap_1d_test.cc

```
00001 #if __cplusplus == 201103L
00002
00003 #include <iostream>
00004
00005 #include "mtk.h"
00006
00007 void Test1() {
00008
00009   mtk::Tools::BeginTestNo(1);
00010
00011   mtk::Lap1D lap2;
00012
00013   bool info = lap2.ConstructLap1D();
00014
00015   if (!info) {
00016     std::cerr << "Mimetic lap (2nd order) could not be built." << std::endl;
00017   }
00018
00019   mtk::Tools::EndTestNo(1);
00020 }
00021
00022 void Test2() {
00023
00024   mtk::Tools::BeginTestNo(2);
00025
00026   mtk::Lap1D lap4;
00027
00028   bool info = lap4.ConstructLap1D(4);
00029
00030   if (!info) {
00031     std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00032   }
00033
00034   mtk::Tools::EndTestNo(2);
00035 }
00036
00037 void Test3() {
00038
00039   mtk::Tools::BeginTestNo(3);
00040
00041   mtk::Lap1D lap6;
00042
00043   bool info = lap6.ConstructLap1D(6);
00044
00045   if (!info) {
00046     std::cerr << "Mimetic lap (6th order) could not be built." << std::endl;
00047   }
00048
00049   mtk::Tools::EndTestNo(3);
00050 }
00051
00052 void Test4() {
00053
00054   mtk::Tools::BeginTestNo(4);
00055
00056   mtk::Lap1D lap8;
00057
00058   bool info = lap8.ConstructLap1D(8);
00059
00060   if (!info) {
00061     std::cerr << "Mimetic lap (8th order) could not be built." << std::endl;
00062   }
00063
00064   mtk::Tools::EndTestNo(4);
00065 }
00066
00067 void Test5() {
00068
00069   mtk::Tools::BeginTestNo(5);
00070
```

```
00071   mtk::Lap1D lap10;
00072
00073   bool info = lap10.ConstructLap1D(10);
00074
00075   if (!info) {
00076     std::cerr << "Mimetic lap (10th order) could not be built." << std::endl;
00077   }
00078
00079   mtk::Tools::EndTestNo(5);
00080 }
00081
00082 void Test6() {
00083
00084   mtk::Tools::BeginTestNo(6);
00085
00086   mtk::Lap1D lap12;
00087
00088   bool info = lap12.ConstructLap1D(12);
00089
00090   if (!info) {
00091     std::cerr << "Mimetic lap (12th order) could not be built." << std::endl;
00092   }
00093
00094   mtk::Tools::EndTestNo(6);
00095 }
00096
00097 void Test7() {
00098
00099   mtk::Tools::BeginTestNo(7);
00100
00101   mtk::Lap1D lap4;
00102
00103   bool info = lap4.ConstructLap1D(4);
00104
00105   if (!info) {
00106     std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00107   }
00108
00109   std::cout << lap4 << std::endl;
00110   std::cout << std::endl;
00111
00112   mtk::Tools::EndTestNo(7);
00113 }
00114
00115 void Test8() {
00116
00117   mtk::Tools::BeginTestNo(8);
00118
00119   mtk::Lap1D lap4;
00120
00121   bool info = lap4.ConstructLap1D(4);
00122
00123   if (!info) {
00124     std::cerr << "Mimetic lap (4th order) could not be built." << std::endl;
00125   }
00126
00127   std::cout << lap4 << std::endl;
00128   std::cout << std::endl;
00129
00130   mtk::UniStgGrid1D aux(0.0, 1.0, 11);
00131
00132   mtk::DenseMatrix lap4_m(lap4.ReturnAsDenseMatrix(aux));
00133
00134   std::cout << lap4_m << std::endl;
00135   std::cout << std::endl;
00136
00137   mtk::Tools::EndTestNo(8);
00138 }
00139
00140 int main () {
00141
00142   std::cout << "Testing MTK 1D Laplacian" << std::endl;
00143
00144   Test1();
00145   Test2();
00146   Test3();
00147   Test4();
00148   Test5();
00149   Test6();
00150   Test7();
00151   Test8();
```

```
00152 }
00153
00154 #else
00155 #include <iostream>
00156 int main () {
00157   std::cout << "This code HAS to be compiled to support C++11." << std::endl;
00158   std::cout << "Exiting..." << std::endl;
00159 }
00160 #endif
```

17.67 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:
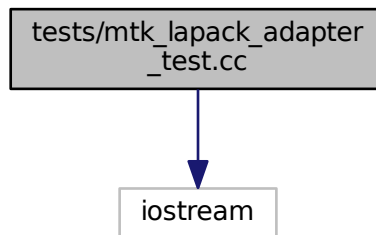
Functions

- int main ()

17.67.1 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.

17.67.2 Function Documentation

17.67.2.1   int main (   )

Definition at line 81 of file mtk_lapack_adapter_test.cc.

17.68   mtk_lapack_adapter_test.cc

```
00001
00010 /*
00011 Copyright (C) 2015, Computational Science Research Center, San Diego State
00012 University. All rights reserved.
00013
00014 Redistribution and use in source and binary forms, with or without modification,
00015 are permitted provided that the following conditions are met:
00016
00017 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00018 and a copy of the modified files should be reported once modifications are
00019 completed. Documentation related to said modifications should be included.
00020
00021 2. Redistributions of source code must be done through direct
00022 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00023
00024 3. Redistributions of source code must retain the above copyright notice, this
00025 list of conditions and the following disclaimer.
00026
00027 4. Redistributions in binary form must reproduce the above copyright notice,
00028 this list of conditions and the following disclaimer in the documentation and/or
00029 other materials provided with the distribution.
00030
00031 5. Usage of the binary form on proprietary applications shall require explicit
00032 prior written permission from the the copyright holders.
00033
00034 6. Neither the name of the copyright holder nor the names of its contributors
00035 may be used to endorse or promote products derived from this software without
00036 specific prior written permission.
00037
00038 The copyright holders provide no reassurances that the source code provided does
00039 not infringe any patent, copyright, or any other intellectual property rights of
00040 third parties. The copyright holders disclaim any liability to any recipient for
00041 claims brought against recipient by any third party for infringement of that
00042 parties intellectual property rights.
00043
00044 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00045 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00046 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00047 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00048 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00049 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00050 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00051 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00052 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00053 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00054 */
00055
00056 #if __cplusplus == 201103L
00057
00058 #include <iostream>
00059 #include <ctime>
00060
00061 #include "mtk.h"
00062
00063 void Test1() {
00064
00065   mtk::Tools::BeginTestNo(1);
00066
00067   mtk::Tools::EndTestNo(1);
00068 }
00069
00070 int main () {
00071
00072   std::cout << "Testing mtk::LAPACKAdapter class." << std::endl;
00073
00074   Test1();
00075 }
00076
00077 #else
00078 #include <iostream>
00079 using std::cout;
00080 using std::endl;
00081 int main () {
00082   cout << "This code HAS to be compiled with support for C++11." << endl;
00083   cout << "Exiting..." << endl;
00084 }
00085 #endif
```

17.69 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:
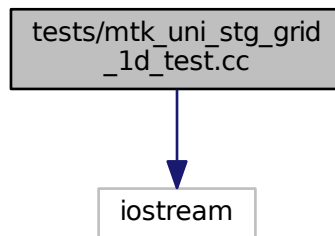
Functions

- int main ()

17.69.1 Detailed Description

Author

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

Definition in file mtk_uni_stg_grid_1d_test.cc.

17.69.2 Function Documentation

17.69.2.1 int main ( )

Definition at line 164 of file mtk_uni_stg_grid_1d_test.cc.

17.70 mtk_uni_stg_grid_1d_test.cc

```
00001
00008 /*
00009 Copyright (C) 2015, Computational Science Research Center, San Diego State
00010 University. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without modification,
00013 are permitted provided that the following conditions are met:
00014
00015 1. Modifications to source code should be reported to: esanchez@mail.sdsu.edu
00016 and a copy of the modified files should be reported once modifications are
00017 completed. Documentation related to said modifications should be included.
```

```
00018
00019 2. Redistributions of source code must be done through direct
00020 downloads from the project's GitHub page: http://www.csrc.sdsu.edu/mtk
00021
00022 3. Redistributions of source code must retain the above copyright notice, this
00023 list of conditions and the following disclaimer.
00024
00025 4. Redistributions in binary form must reproduce the above copyright notice,
00026 this list of conditions and the following disclaimer in the documentation and/or
00027 other materials provided with the distribution.
00028
00029 5. Usage of the binary form on proprietary applications shall require explicit
00030 prior written permission from the the copyright holders.
00031
00032 6. Neither the name of the copyright holder nor the names of its contributors
00033 may be used to endorse or promote products derived from this software without
00034 specific prior written permission.
00035
00036 The copyright holders provide no reassurances that the source code provided does
00037 not infringe any patent, copyright, or any other intellectual property rights of
00038 third parties. The copyright holders disclaim any liability to any recipient for
00039 claims brought against recipient by any third party for infringement of that
00040 parties intellectual property rights.
00041
00042 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
00043 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
00044 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00045 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
00046 ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
00047 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
00048 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
00049 ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00050 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
00051 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00052 */
00053
00054 #if __cplusplus == 201103L
00055
00056 #include <iostream>
00057 #include <ctime>
00058
00059 #include "mtk.h"
00060
00061 void Test1() {
00062
00063   mtk::Tools::BeginTestNo(1);
00064
00065   mtk::UniStgGrid1D gg;
00066
00067   std::cout << gg << std::endl;
00068
00069   mtk::Tools::EndTestNo(1);
00070 }
00071
00072 mtk::Real ScalarFieldOne(mtk::Real xx) {
00073
00074   return 2.0*xx;
00075 }
00076
00077 void Test2() {
00078
00079   mtk::Tools::BeginTestNo(2);
00080
00081   mtk::Real aa = 0.0;
00082   mtk::Real bb = 1.0;
00083
00084   int nn = 5;
00085
00086   mtk::UniStgGrid1D gg(aa, bb, nn);
00087
00088   std::cout << gg << std::endl;
00089
00090   gg.BindScalarField(ScalarFieldOne);
00091
00092   std::cout << gg << std::endl;
00093
00094   mtk::Tools::EndTestNo(2);
00095 }
00096
00097 void Test3() {
00098
```

```
00099    mtk::Tools::BeginTestNo(3);
00100
00101    mtk::Real aa = 0.0;
00102    mtk::Real bb = 1.0;
00103
00104    int nn = 5;
00105
00106    mtk::UniStgGrid1D gg(aa, bb, nn);
00107
00108    std::cout << gg << std::endl;
00109
00110    gg.BindScalarField(ScalarFieldOne);
00111
00112    std::cout << gg << std::endl;
00113
00114    if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_03.dat", "x", "u(x)")) {
00115      std::cerr << "Error writing to file." << std::endl;
00116    }
00117
00118    mtk::Tools::EndTestNo(3);
00119 }
00120
00121 mtk::Real VectorFieldXComponentOne(mtk::Real xx) {
00122
00123    return xx*xx;
00124 }
00125
00126 void Test4() {
00127
00128    mtk::Tools::BeginTestNo(4);
00129
00130    mtk::Real aa = 0.0;
00131    mtk::Real bb = 1.0;
00132
00133    int nn = 20;
00134
00135    mtk::UniStgGrid1D gg(aa, bb, nn, mtk::VECTOR);
00136
00137    std::cout << gg << std::endl;
00138
00139    gg.BindVectorField(VectorFieldXComponentOne);
00140
00141    std::cout << gg << std::endl;
00142
00143    if(!gg.WriteToFile("mtk_uni_stg_grid_1d_test_04.dat", "x", "v(x)")) {
00144      std::cerr << "Error writing to file." << std::endl;
00145    }
00146
00147    mtk::Tools::EndTestNo(4);
00148 }
00149
00150 int main () {
00151
00152    std::cout << "Testing mtk::UniStgGrid1D class." << std::endl;
00153
00154    Test1();
00155    Test2();
00156    Test3();
00157    Test4();
00158 }
00159
00160 #else
00161 #include <iostream>
00162 using std::cout;
00163 using std::endl;
00164 int main () {
00165    cout << "This code HAS to be compiled with support for C++11." << endl;
00166    cout << "Exiting..." << endl;
00167 }
00168 #endif
```

Index