AMG Methods

Generated by Doxygen 1.8.11

Contents

Chapter 1

Introduction

1.1 Description

This program aim to solve linear systems arising from conforming and discontinuous finite element discretizions with algebraic multigrid methods.
All source and header files are written in C++11 language.

1.2 Dependencies

Software needs that on system must be installed the following dependencies :

- `CMake` (version 3.5.1 or above), cross-platform family of tools designed to build, test and package software;
- `Make` (version 4.1 or above), a tool which controls the generation of executables of a program from the program's source files;
- `GCC` (version 5.4.0 or above), GNU Compiler Collection;
- `Eigen` (version 3.3 or above), library for linear algebra: matrices, vectors, numerical solvers, and related algorithms;

We also use the following library, provided in folder *include/*:

- `GetPot` (v. 1.1.17), it is used for parsing comand line arguments and configuration files.

1.3 Compile

To generate the executable it is provided file *CMakeLists.txt* (in top-level folder).

Create a compilation folder and open it with the following commands :

```
$ mkdir build
$ cd build
```

Now the system is ready for the configuration :

```
$ cmake ..
```

If the Eigen library folder is not installed in a system one then cmake will give an error message, therefore it is necessary to specify the folder where the library is installed with the following command :

```
$ cmake .. -DEIGEN3_INCLUDE_DIR=path_folder/name_folder
```

Finally :

```
$ make
```

will create the executable *main*.

1.4 Parameter configuration

Note

By default, configuration file is saved in the same folder of CMakeLists.txt.

Before running the program configuration file must be set (default: *config.pot*). In configuration file it can be possible modify some problem parameters, in *config.pot* file all details of parameters are explained.
For example it can be possible decide type of mu-cycle, how to use AMG (stand-alone or preconditioner), and more others.

1.5 Run

In order to use predefined configuration file (*config.pot*), move into the executable folder (folder *build/*) and digit :

```
$ ./main
```

To specify a different configuration file run main program in the following ways:

```
$ ./main -f configuration_directory_filename
```

or:

```
$ ./main --file configuration_directory_filename
```

At the end of the program the results can be print on screen or they can be saved on files in the choosen output directory, specified in configuration file.

1.6 Examples

In folder *share/examples/* are given test files (matrices and associated right-hand side vectors have the same name of files except for "flag" A or f), this folder will be then copied in compilation folder *build/* with instruction in *CMake↩ Lists.txt*, i.e. the examples will be copied in folder *build/share/examples* and therefore the results will be saved (if not declared differently by the user) in folder *build/share/results*.

The name of test files give to the user some important information, therefore we detail the name of files.

The following text denotes the name of a matrix (initial "flag" A),

A_FEM_level_h_fem_Pp_TS.txt

whereas the following text denotes the name of the associated right-hand side vector with matrix A (initial "flag" f),

f_FEM_level_h_fem_Pp_TS.txt

The FEM "flag" denotes the type of finite element discretizations (CG: conforming Galerkin, DG: discontinouous Galerkin), the value h is a "flag" for the grid parameter (bigger h stands for finer refinement of the meshes) and the value p is the degree of approximate polynomials. Finally the flag TS denotes that the mesh is a structured simplicial triangular one.

1.7 Documentation

If `Doxygen` and `GraphViz` are installed on system, the following instruction (runned in the top-level folder) will create the documentation in folder *doc/*

```
$ doxygen Doxyfile.in
```

then move into the folder *doc/latex/* to create the pdf file with the following instructions

```
$ cd doc/latex/
$ make
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

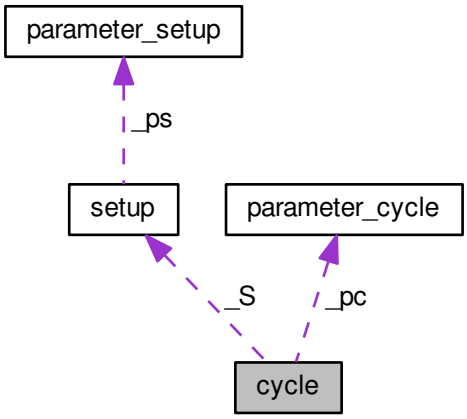
Class Documentation

5.1 cycle Class Reference

This class defines one iteration of mu-cycle.

`#include <cycle.h>`

Collaboration diagram for cycle:



Public Member Functions

- cycle ()=default

  Constructor (defaulted)
- cycle (const setup &S, const Vec &f, const parameter_cycle &p)

  Constructor.
- ~cycle ()

  Destructor (defaulted)

- const setup & get_S () const

  Reading setup S.
- const Vec & get_u (const size_t &n)

  Reading vector u.
- void set_u (const size_t &n, const Vec &v)

  Writing vector u.
- const Vec & get_f (const size_t &n)

  Reading vector f.
- void set_f (const size_t &n, const Vec &v)

  Writing vector f.
- void Cycle (int lev)

  Cycle iteration.

Private Member Functions

- void GS (Vec &u, const Vec &f, const int &j, const int &maxit)

  Gauss-Seidel method.

Private Attributes

- setup _S

  setup containing coarser matrices and interpolation operators
- vector< Vec > _f

  vector of right-hand side on all levels
- vector< Vec > _u

  vector of solution on all levels
- parameter_cycle _pc

  parameters of cycle

5.1.1 Detailed Description

This class defines one iteration of mu-cycle.

Definition at line 24 of file cycle.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 cycle (const setup & *S*, const Vec & *f*, const parameter_cycle & *p*)

Constructor.

Parameters

in	*S*	setup containing coarser matrices and interpolation operators
in	*f*	right-hand side on finest level
in	*p*	parameters of cycle

Definition at line 14 of file cycle.cpp.

5.1.3 Member Function Documentation

5.1.3.1 const setup& get_S ( ) const [inline]

Reading setup S.

Parameters

out	*S*	setup containing coarser matrices and interpolation operators

Definition at line 58 of file cycle.h.

5.1.3.2 const Vec& get_u (const size_t & *n*) [inline]

Reading vector u.

Parameters

in	*n*	current level of matrix/vector
out	*u[n]*	vector u at current level

Definition at line 70 of file cycle.h.

5.1.3.3 void set_u (const size_t & *n,* const Vec & *v*) [inline]

Writing vector u.

Parameters

in	*n*	current level of matrix/vector
in	*v*	vector to be copied
out	*u[n]*	assigned vector u at current level

Definition at line 87 of file cycle.h.

5.1.3.4 const Vec& get_f (const size_t & *n*) [inline]

Reading vector f.

Parameters

in	*n*	current level of matrix/vector
out	*f[n]*	vector f at current level

Definition at line 103 of file cycle.h.

5.1.3.5 void set_f (const size_t & *n*, const Vec & *v*) [inline]

Writing vector f.

Parameters

in	*n*	current level of matrix/vector
in	*v*	vector to be copied
out	*f[n]*	assigned vector f at current level

Definition at line 120 of file cycle.h.

5.1.3.6 void Cycle (int *lev*)

Cycle iteration.

Parameters

in	*lev*	current level of coarser matrices, 0 is for finest level

Definition at line 40 of file cycle.cpp.

5.1.3.7 void GS (Vec & *u*, const Vec & *f*, const int & *j*, const int & *maxit*) [private]

Gauss-Seidel method.

Parameters

in	*u*	initial solution guess
in	*j*	current level of matrix/vector
in	*f*	right-hand side on j level
in	*maxit*	maximum number of smoothing iterations

Definition at line 24 of file cycle.cpp.

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

- laura/AMG_Methods/include/cycle.h
- laura/AMG_Methods/src/cycle.cpp

5.2 GetPot Class Reference

This class read input values from files (library GetPot http://getpot.sourceforge.net).

`#include <GetPot.h>`

Classes

- struct variable

Public Member Functions

- **GetPot** (const GetPot &)
- **GetPot** (const int argc_, char **argv_, const char *FieldSeparator=0x0)
- **GetPot** (const char *FileName, const char *CommentStart=0x0, const char *CommentEnd=0x0, const char *FieldSeparator=0x0)
- GetPot & **operator=** (const GetPot &)
- void **absorb** (const GetPot &That)
- void **clear_requests** ()
- void **disable_request_recording** ()
- void **enable_request_recording** ()
- const std::string **operator[ ]** (unsigned Idx) const
- int **get** (unsigned Idx, int Default) const
- double **get** (unsigned Idx, const double &Default) const
- const std::string **get** (unsigned Idx, const char *Default) const
- unsigned **size** () const
- bool **options_contain** (const char *FlagList) const
- bool **argument_contains** (unsigned Idx, const char *FlagList) const
- int **operator()** (const char *VarName, int Default) const
- double **operator()** (const char *VarName, const double &Default) const
- const std::string **operator()** (const char *VarName, const char *Default) const
- int **operator()** (const char *VarName, int Default, unsigned Idx) const
- double **operator()** (const char *VarName, const double &Default, unsigned Idx) const
- const std::string **operator()** (const char *VarName, const char *Default, unsigned Idx) const
- void **set** (const char *VarName, const char *Value, const bool Requested=true)
- void **set** (const char *VarName, const double &Value, const bool Requested=true)
- void **set** (const char *VarName, const int Value, const bool Requested=true)
- unsigned **vector_variable_size** (const char *VarName) const
- STRING_VECTOR **get_variable_names** () const
- STRING_VECTOR **get_section_names** () const
- void **set_prefix** (const char *Prefix)
- bool **search_failed** () const
- void **disable_loop** ()
- void **enable_loop** ()
- void **reset_cursor** ()
- void **init_multiple_occurrence** ()
- bool **search** (const char *option)
- bool **search** (unsigned No, const char *P,...)
- int **next** (int Default)
- double **next** (const double &Default)
- const std::string **next** (const char *Default)
- int **follow** (int Default, const char *Option)
- double **follow** (const double &Default, const char *Option)
- const std::string **follow** (const char *Default, const char *Option)
- int **follow** (int Default, unsigned No, const char *Option,...)
- double **follow** (const double &Default, unsigned No, const char *Option,...)
- const std::string **follow** (const char *Default, unsigned No, const char *Option,...)
- std::vector< std::string > **nominus_followers** (const char *Option)
- std::vector< std::string > **nominus_followers** (unsigned No,...)
- int **direct_follow** (int Default, const char *Option)

- double **direct_follow** (const double &Default, const char *Option)
- const std::string **direct_follow** (const char *Default, const char *Option)
- std::vector< std::string > **string_tails** (const char *StartString)
- std::vector< int > **int_tails** (const char *StartString, const int Default=1)
- std::vector< double > **double_tails** (const char *StartString, const double Default=1.0)
- STRING_VECTOR **nominus_vector** () const
- unsigned **nominus_size** () const
- std::string **next_nominus** ()
- STRING_VECTOR **unidentified_arguments** (unsigned Number, const char *Known,...) const
- STRING_VECTOR **unidentified_arguments** (const STRING_VECTOR &Knowns) const
- STRING_VECTOR **unidentified_arguments** () const
- STRING_VECTOR **unidentified_options** (unsigned Number, const char *Known,...) const
- STRING_VECTOR **unidentified_options** (const STRING_VECTOR &Knowns) const
- STRING_VECTOR **unidentified_options** () const
- std::string **unidentified_flags** (const char *Known, int ArgumentNumber) const
- STRING_VECTOR **unidentified_variables** (unsigned Number, const char *Known,...) const
- STRING_VECTOR **unidentified_variables** (const STRING_VECTOR &Knowns) const
- STRING_VECTOR **unidentified_variables** () const
- STRING_VECTOR **unidentified_sections** (unsigned Number, const char *Known,...) const
- STRING_VECTOR **unidentified_sections** (const STRING_VECTOR &Knowns) const
- STRING_VECTOR **unidentified_sections** () const
- STRING_VECTOR **unidentified_nominuses** (unsigned Number, const char *Known,...) const
- STRING_VECTOR **unidentified_nominuses** (const STRING_VECTOR &Knowns) const
- STRING_VECTOR **unidentified_nominuses** () const
- int **print** () const

Private Member Functions

- void **__basic_initialization** ()
- void **__record_argument_request** (const std::string &Arg)
- void **__record_variable_request** (const std::string &Arg)
- void **__set_variable** (const char *VarName, const char *Value)
- void **__parse_argument_vector** (const STRING_VECTOR &ARGV)
- const variable * **__find_variable** (const char *) const
- const char * **__match_starting_string** (const char *StartString)
- bool **__check_flags** (const std::string &Str, const char *FlagList) const
- int **__convert_to_type** (const std::string &String, int Default) const
- double **__convert_to_type** (const std::string &String, double Default) const
- const std::string **__get_remaining_string** (const std::string &String, const std::string &Start) const
- bool **__search_string_vector** (const STRING_VECTOR &Vec, const std::string &Str) const
- void **__skip_whitespace** (std::istream &istr)
- const std::string **__get_next_token** (std::istream &istr)
- const std::string **__get_string** (std::istream &istr)
- const std::string **__get_until_closing_bracket** (std::istream &istr)
- STRING_VECTOR **__read_in_stream** (std::istream &istr)
- STRING_VECTOR **__read_in_file** (const char *FileName)
- std::string **__process_section_label** (const std::string &Section, STRING_VECTOR &section_stack)
- std::string **__DBE_expand_string** (const std::string str)
- std::string **__DBE_expand** (const std::string str)
- const GetPot::variable * **__DBE_get_variable** (const std::string str)
- STRING_VECTOR **__DBE_get_expr_list** (const std::string str, const unsigned ExpectedNumber)
- std::string **__double2string** (const double &Value) const
- std::string **__int2string** (const int &Value) const
- STRING_VECTOR **__get_section_tree** (const std::string &FullPath)

Private Attributes

- std::string **prefix**
- std::string **section**
- STRING_VECTOR **section_list**
- STRING_VECTOR **argv**
- unsigned **cursor**
- bool **search_loop_f**
- bool **search_failed_f**
- int **nominus_cursor**
- std::vector< unsigned > **idx_nominus**
- std::vector< variable > **variables**
- std::string **_comment_start**
- std::string **_comment_end**
- std::string **_field_separator**
- std::vector< char * > **__internal_string_container**
- STRING_VECTOR **_requested_arguments**
- STRING_VECTOR **_requested_variables**
- STRING_VECTOR **_requested_sections**
- bool **__request_recording_f**

5.2.1 Detailed Description

This class read input values from files (library GetPot `http://getpot.sourceforge.net`).

Definition at line 82 of file GetPot.h.

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

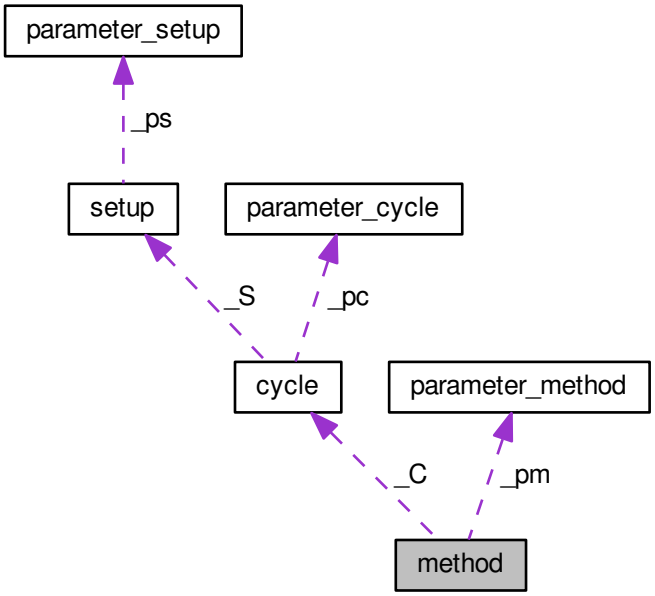
- laura/AMG_Methods/include/GetPot.h

5.3 method Class Reference

This class defines AMG methods: AMG stand-alone or PCG preconditioned conjugate gradient.

```
#include <method.h>
```

Collaboration diagram for method:



Public Member Functions

- method ()=default

  Constructor (defaulted)
- method (cycle &C, const parameter_method &p)

  Constructor.
- ~method ()

  Destructor (defaulted)
- void AMGCycle ()

  AMG stand-alone method.
- void PCGCycle ()

  PCG method (AMG as preconditioner)
- const int & get_iter () const

  Reading number of iterations to achieve convergence.
- const bool & get_flag () const

  Reading flag.
- const Real & get_rho () const

  Reading convergence factor.
- const Vec & get_solution () const

  Reading solution vector.

Private Attributes

- cycle _C
  definition of one iteration of mu-cycle
- Vec _solution
  vector containing solution
- int _iter
  number of iterations to achieve convergence
- Real _rho
  convergence factor of method
- bool _flag
  flag associated with convergence of method (0 convergence, 1 otherwise)
- parameter_method _pm
  parameters of method

5.3.1 Detailed Description

This class defines AMG methods: AMG stand-alone or PCG preconditioned conjugate gradient.

Definition at line 24 of file method.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 method (cycle & *C*, const parameter_method & *p*)

Constructor.

Parameters

in	*C*	definition of one iteration of mu-cycle
in	*p*	parameters of method

Definition at line 14 of file method.cpp.

5.3.3 Member Function Documentation

5.3.3.1 const int& get_iter ( ) const [inline]

Reading number of iterations to achieve convergence.

Parameters

out	*iter*	number of iterations to achieve convergence

Definition at line 71 of file method.h.

5.3.3.2 const bool& get_flag ( ) const [inline]

Reading flag.

Parameters

out	*flag*	flag associated with convergence of method (0 convergence, 1 otherwise)

Definition at line 82 of file method.h.

5.3.3.3 const Real& get_rho ( ) const [inline]

Reading convergence factor.

Parameters

out	*rho*	convergence factor of method

Definition at line 93 of file method.h.

5.3.3.4 const Vec& get_solution ( ) const [inline]

Reading solution vector.

Parameters

out	*solution*	vector containing solution

Definition at line 104 of file method.h.

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

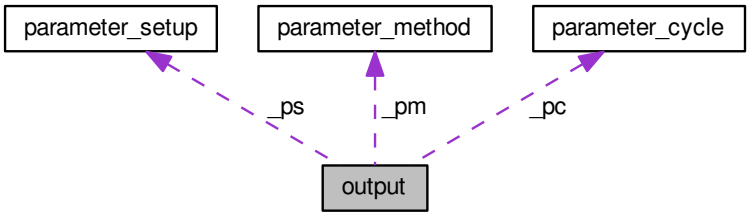
- laura/AMG_Methods/include/method.h
- laura/AMG_Methods/src/method.cpp

5.4 output Class Reference

This class contains the printing and saving tools.

`#include <output.h>`

Collaboration diagram for output:



Public Member Functions

- output ()=default

  Constructor (defaulted)
- output (const string &testname, const string &inputA, const string &inputf, const string &fem, const string &method, const int &iter, const Real &rho, const bool &flag, const parameter_setup &ps, const parameter_cycle &pc, const parameter_method &pm)

  Constructor.
- ~output ()

  Destructor (defaulted)
- void print_on_screen () const

  Print results on screen.
- void print_on_file (const string &directory) const

  Print results on file.

Private Attributes

- string _testname

  name of output file
- string _inputA

  name of input file for matrix
- string _inputf

  name of input file for vector
- string _fem

  type of finite element discretization (CG conforming Galerkin, DG discontinuous Galerkin)
- string _method

  type of AMG method (AMG as stand-alone, PCG as preconditioner for conjugate gradient)
- int _iter

  number of iterations to achieve convergence
- bool _flag

  flag associated with convergence (0 convergence, 1 otherwise)
- Real _rho

  convergence factor
- parameter_setup _ps

  parameters of setup
- parameter_cycle _pc

  parameters of cycle
- parameter_method _pm

  parameters of method

5.4.1 Detailed Description

This class contains the printing and saving tools.

Definition at line 25 of file output.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 output (const string & *testname,* const string & *inputA,* const string & *inputf,* const string & *fem,* const string & *method,* const int & *iter,* const Real & *rho,* const bool & *flag,* const parameter_setup & *ps,* const parameter_cycle & *pc,* const parameter_method & *pm*)

Constructor.

Parameters

in	*testname*	name of output file
in	*inputA*	name of input file for matrix
in	*inputf*	name of input file for vector
in	*fem*	type of finite element discretization (CG conforming Galerkin, DG discontinuous Galerkin)
in	*method*	type of AMG method (AMG as stand-alone, PCG as preconditioner for conjugate gradient)
in	*iter*	number of iterations to achieve convergence
in	*rho*	convergence factor
in	*flag*	flag associated with convergence (0 convergence, 1 otherwise)
in	*parameter_setup*	parameters of setup
in	*parameter_cycle*	parameters of cycle
in	*parameter_method*	parameters of method

Definition at line 14 of file output.cpp.

5.4.3 Member Function Documentation

5.4.3.1 void print_on_file (const string & *directory*) const

Print results on file.

Parameters

in	*directory*	file location path

Definition at line 58 of file output.cpp.

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

- laura/AMG_Methods/include/output.h
- laura/AMG_Methods/src/output.cpp

5.5 parameter_cycle Class Reference

This class contains cycle parameters.

`#include <parameter_cycle.h>`

Public Member Functions

- parameter_cycle ()=default
  Constructor (defaulted)
- parameter_cycle (const int &nlevel, const int &nu1, const int &nu2, const int &gamma)
  Constructor.
- ~parameter_cycle ()
  Destructor (defaulted)
- const int & get_nlevel () const
  Reading parameter nlevel.
- const int & get_nu1 () const
  Reading parameter nu1.
- const int & get_nu2 () const
  Reading parameter nu2.
- const int & get_mu () const
  Reading parameter mu.

Private Attributes

- int _nlevel
  number of coarser levels
- int _nu1
  number of pre-smoothing iterations
- int _nu2
  number of post-smoothing iterations
- int _mu
  flag to decide type of cycle: mu=1 V-cycle, mu=2 W-cycle

5.5.1 Detailed Description

This class contains cycle parameters.

Definition at line 23 of file parameter_cycle.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 parameter_cycle (const int & *nlevel*, const int & *nu1*, const int & *nu2*, const int & *gamma*)

Constructor.

Parameters

in	*nlevel*	number of coarser levels
in	*nu1*	number of pre-smoothing iterations
in	*nu2*	number of post-smoothing iterations
in	*mu*	flag to decide type of cycle: mu=1 V-cycle, mu=2 W-cycle

Definition at line 14 of file parameter_cycle.cpp.

5.5.3 Member Function Documentation

5.5.3.1 const int& get_nlevel ( ) const [inline]

Reading parameter nlevel.

Parameters

out	*nlevel*	number of coarser levels

Definition at line 58 of file parameter_cycle.h.

5.5.3.2 const int& get_nu1 ( ) const [inline]

Reading parameter nu1.

Parameters

out	*nu1*	number of pre-smoothing iterations

Definition at line 69 of file parameter_cycle.h.

5.5.3.3 const int& get_nu2 ( ) const [inline]

Reading parameter nu2.

Parameters

out	*nu2*	number of post-smoothing iterations

Definition at line 80 of file parameter_cycle.h.

5.5.3.4 const int& get_mu ( ) const [inline]

Reading parameter mu.

Parameters

out	*mu*	flag to decide type of cycle: mu=1 V-cycle, mu=2 W-cycle

Definition at line 91 of file parameter_cycle.h.

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

- laura/AMG_Methods/include/parameter_cycle.h
- laura/AMG_Methods/src/parameter_cycle.cpp

5.6 parameter_method Class Reference

This class contains method parameters.

`#include <parameter_method.h>`

Public Member Functions

- parameter_method ()=default
  Constructor (defaulted)
- parameter_method (const Real &tol, const int &maxiter)
  Constructor.
- ~parameter_method ()
  Destructor (defaulted)
- const int & get_maxiter () const
  Reading parameter maxiter.
- const Real & get_tol () const
  Reading parameter tol.

Private Attributes

- Real _tol
  tolerance
- int _maxiter
  number of maximum iterations

5.6.1 Detailed Description

This class contains method parameters.

Definition at line 23 of file parameter_method.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 parameter_method (const Real & *tol*, const int & *maxiter*)

Constructor.

Parameters

in	*tol*	tolerance
in	*maxiter*	number of maximum iterations

Definition at line 14 of file parameter_method.cpp.

5.6.3 Member Function Documentation

5.6.3.1 const int& get_maxiter ( ) const [inline]

Reading parameter maxiter.

Parameters

out	*maxiter*	number of maximum iterations

Definition at line 56 of file parameter_method.h.

5.6.3.2 const Real& get_tol ( ) const [inline]

Reading parameter tol.

Parameters

out	*tol*	tolerance

Definition at line 67 of file parameter_method.h.

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

- laura/AMG_Methods/include/parameter_method.h
- laura/AMG_Methods/src/parameter_method.cpp

5.7 parameter_setup Class Reference

This class contains setup parameters.

`#include <parameter_setup.h>`

Public Member Functions

- parameter_setup ()=default
  Constructor (defaulted)
- parameter_setup (const int &nmatrix, const Real &theta)
  Constructor.
- ~parameter_setup ()
  Destructor (defaulted)
- const int & get_nmatrix () const
  Reading parameter nmatrix.
- const Real & get_theta () const
  Reading parameter theta.

Private Attributes

- int _nmatrix
  number of coarser matrices
- Real _theta
  strong connection threshold

5.7.1 Detailed Description

This class contains setup parameters.

Definition at line 23 of file parameter_setup.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 parameter_setup (const int & *nmatrix*, const Real & *theta*)

Constructor.

Parameters

in	*nmatrix*	number of coarser matrices
in	*theta*	strong connection threshold

Definition at line 14 of file parameter_setup.cpp.

5.7.3 Member Function Documentation

5.7.3.1 const int& get_nmatrix ( ) const [inline]

Reading parameter nmatrix.

Parameters

out	*nmatrix*	number of coarser matrices

Definition at line 56 of file parameter_setup.h.

5.7.3.2 const Real& get_theta ( ) const [inline]

Reading parameter theta.

Parameters

out	*theta*	strong connection threshold

Definition at line 67 of file parameter_setup.h.

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

- laura/AMG_Methods/include/parameter_setup.h
- laura/AMG_Methods/src/parameter_setup.cpp

5.8 sets Class Reference

This class performs some properties and utilities of mathematical sets.

`#include <sets.h>`

Public Member Functions

- sets ()=default
  Constructor (defaulted)
- sets (const size_t &dim)
  Constructor.
- sets (const sets &A)
  Copy constructor.
- ~sets ()
  Destructor (defaulted)
- int & operator[ ] (const size_t &n)
  Definition of operator [], writing version.
- const int & operator[ ] (const size_t &n) const
  Definition of operator [], reading version.
- void addElement (const int &s)
  Add an element in the set.
- void deleteElement (const int &s)
  Delete an element in the set.
- bool isMember (const int &s)

Check if an element is in the set.

- bool isEmpty ()

  Check if a set is empty.

- int find_pos_set (const int &s)

  Find the position of an element in the set.

- int cardinality ()

  Cardinality of the set.

- void sort_set ()

  Reorder the set.

- void clear_set ()

  Delete all element of the set.

Static Public Member Functions

- static sets union_set (sets &A, sets &B)

  Union between two sets.

- static sets diff_set (sets &A, sets &B)

  Difference between two sets.

- static sets inter_set (sets &A, sets &B)

  Intersection between two sets.

Private Attributes

- vector< int > _set

  definition of set

5.8.1 Detailed Description

This class performs some properties and utilities of mathematical sets.

Definition at line 23 of file sets.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 sets (const size_t & *dim*)

Constructor.

Parameters

in	*dim*	cardinality of the set

Definition at line 14 of file sets.cpp.

5.8.2.2 sets (const sets & *A*)

Copy constructor.

Parameters

in	*A*	set to be copied

Definition at line 16 of file sets.cpp.

5.8.3 Member Function Documentation

5.8.3.1 int& operator[ ] (const size_t & *n*) [inline]

Definition of operator [], writing version.

Parameters

in	*n*	access position to an element of the set
out	*set[n]*	write element in the choosen position of the set

Definition at line 64 of file sets.h.

5.8.3.2 const int& operator[ ] (const size_t & *n*) const [inline]

Definition of operator [], reading version.

Parameters

in	*n*	access position to an element of the set
out	*set[n]*	read element in the choosen position of the set

Definition at line 79 of file sets.h.

5.8.3.3 void addElement (const int & *s*)

Add an element in the set.

Parameters

in	*s*	element to be added

Definition at line 28 of file sets.cpp.

5.8.3.4 void deleteElement (const int & *s*)

Delete an element in the set.

Parameters

in	*s*	element to be deleted

Definition at line 40 of file sets.cpp.

5.8.3.5 bool isMember (const int & *s*)

Check if an element is in the set.

Parameters

in	*s*	element to be found
out	*0,1*	: 1 if s is in the set, 0 otherwise

Definition at line 18 of file sets.cpp.

5.8.3.6 bool isEmpty ( )

Check if a set is empty.

Parameters

out	*0,1*	: 1 if the set is empty, 0 otherwise

Definition at line 80 of file sets.cpp.

5.8.3.7 int find_pos_set (const int & *s*)

Find the position of an element in the set.

Parameters

in	*s*	element to be found
out	*d*	: position of the element

Definition at line 33 of file sets.cpp.

5.8.3.8 sets union_set (sets & *A*, sets & *B*) [static]

Union between two sets.

Parameters

out	*U*	: union set between A and B
in	*A,B*	: two sets

Definition at line 50 of file sets.cpp.

5.8.3.9 sets diff_set (sets & *A*, sets & *B*) [static]

Difference between two sets.

Parameters

out	*D*	: difference set between A and B (D=A-B)
in	*A,B*	: two sets

Definition at line 60 of file sets.cpp.

5.8.3.10 sets inter_set (sets & *A*, sets & *B*) [static]

Intersection between two sets.

Parameters

out	*I*	: intersection set between A and B
in	*A,B*	: two sets

Definition at line 70 of file sets.cpp.

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

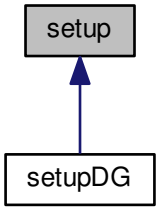
- laura/AMG_Methods/include/sets.h
- laura/AMG_Methods/src/sets.cpp

5.9 setup Class Reference

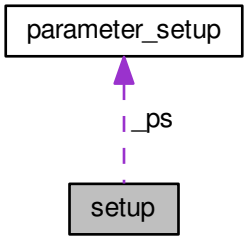
This class defines the construction of coarser matrices and interpolation operators, in particular for matrices stemming from conforming Galerkin discretization.

```
#include <setup.h>
```

Inheritance diagram for setup:



Collaboration diagram for setup:



Public Member Functions

- setup ()=default
  Constructor (defaulted)
- setup (const SpMat &A, const parameter_setup &p)
  Constructor.
- ~setup ()
  Destructor (defaulted)
- const SpMat & get_A (const size_t &n) const
  Reading matrix A.
- const SpMat & get_I (const size_t &n) const
  Reading interpolation operator I.

Protected Member Functions

- void strong_influence_dependence (const SpMat &A, vector< sets > &S, vector< sets > &St, vector< sets > &Dw)
  Definition of strong connections.

- void colouring_scheme (vector< sets > &S, vector< sets > &St, sets &C, sets &F)

  First step of coarsening strategy: C/F splitting.
- void coarse_strong_dependence (vector< sets > &S, vector< sets > &Ci, vector< sets > &Ds, sets C)

  Definition of vectors of coarse-interpolatory sets and of strong non-interpolatory sets.
- void check_modify (sets &C, sets &F, vector< sets > &Ci, vector< sets > &Ds)

  Second step of coarsening strategy: C/F splitting.
- void interpolation (const SpMat &A, SpMat &I, sets &C, const vector< sets > &Ci, const vector< sets > &Ds, const vector< sets > &Dw)

  Interpolation formula.
- void CG_setup ()

  Construnction of coarser matrices and interpolation operators for matrix stemming from conforming Galerkin discretization.
- vector< Real > element_set (const SpMat &A, sets &B, const int &c)

  Utility:
- void minus_maxrow_maxcol (const SpMat &A, vector< Real > &maxrow, vector< Real > &maxcol)

  Utility:

Protected Attributes

- vector< SpMat > _A

  vector containing coarser matrices
- vector< SpMat > _I

  vector containing interpolation operators
- parameter_setup _ps

  parameters of setup

5.9.1 Detailed Description

This class defines the construction of coarser matrices and interpolation operators, in particular for matrices stemming from conforming Galerkin discretization.

Definition at line 24 of file setup.h.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 setup (const SpMat & *A*, const parameter_setup & *p*)

Constructor.

Parameters

in	*A*	input matrix defined on finest level
in	*p*	parameters of setup

Definition at line 15 of file setup.cpp.

5.9.3 Member Function Documentation

5.9.3.1 const SpMat& get_A (const size_t & *n*) const [inline]

Reading matrix A.

Parameters

in	*n*	current position of coarser matrix
out	*A[n]*	matrix A at current position

Definition at line 58 of file setup.h.

5.9.3.2 const SpMat& get_I (const size_t & *n*) const [inline]

Reading interpolation operator I.

Parameters

in	*n*	current position of interpolation operator
out	*I[n]*	operator I at current position

Definition at line 73 of file setup.h.

5.9.3.3 void strong_influence_dependence (const SpMat & *A*, vector< sets > & *S*, vector< sets > & *St*, vector< sets > & *Dw*) [protected]

Definition of strong connections.

Parameters

in	*A*	input matrix defined on finest level
in	*S*	initialization of vector of sets containing all strong dependence connections (it will be built in the method)
in	*St*	initialization of vector of sets containing all strong influence conncetions (it will be built in the method)
in	*Dw*	initialization of vector of sets containing all weak connections (it will be built in the method)

Definition at line 42 of file setup.cpp.

5.9.3.4 void colouring_scheme (vector< sets > & *S*, vector< sets > & *St*, sets & *C*, sets & *F*) [protected]

First step of coarsening strategy: C/F splitting.

Parameters

in	*S*	vector of sets containing all strong dependence connections
in	*St*	vector of sets containing all strong influence conncetions
in	*C*	initialization of C-points (it will be built in the method)
in	*F*	initialization of F-points (it will be built in the method)

Definition at line 84 of file setup.cpp.

5.9.3.5 void coarse_strong_dependence (vector< sets > & *S*, vector< sets > & *Ci*, vector< sets > & *Ds*, sets *C*) [protected]

Definition of vectors of coarse-interpolatory sets and of strong non-interpolatory sets.

Parameters

in	*S*	vector of sets containing all strong dependence connections
in	*Ci*	initialization of vector of coarse interpolatory sets (it will be built in the method)
in	*Ds*	initialization of vector of strong non-interpolatory sets (it will be built in the method)
in	*C*	C-points of C/F-splitting

Definition at line 128 of file setup.cpp.

5.9.3.6 void check_modify (sets & *C*, sets & *F*, vector< sets > & *Ci*, vector< sets > & *Ds*) [protected]

Second step of coarsening strategy: C/F splitting.

Parameters

in	*C*	C-points of C/F-splitting
in	*F*	F-points of C/F-splitting
in	*Ci*	vector of coarse interpolatory sets
in	*Ds*	vector of strong non-interpolatory sets

Definition at line 145 of file setup.cpp.

5.9.3.7 void interpolation (const SpMat & *A*, SpMat & *I*, sets & *C*, const vector< sets > & *Ci*, const vector< sets > & *Ds*, const vector< sets > & *Dw*) [protected]

Interpolation formula.

Parameters

in	*A*	input matrix defined on finest level
in	*I*	initialization of interpolation operator (it will be built in the method)
in	*C*	C-points of C/F-splitting

Parameters

in	*Ci*	vector of coarse interpolatory sets
in	*Ds*	vector of strong non-interpolatory sets
in	*Dw*	vector of weak non-interpolatory sets

Definition at line 179 of file setup.cpp.

5.9.3.8 vector< Real > element_set (const SpMat & *A*, sets & *B*, const int & *c*) [protected]

Utility:

Parameters

in	*A*	input matrix defined on finest level
in	*B*	indices set
in	*c*	index
out	*Aeval*	vector containing all values of A(B,c)

Definition at line 292 of file setup.cpp.

5.9.3.9 void minus_maxrow_maxcol (const SpMat & *A*, vector< Real > & *maxrow*, vector< Real > & *maxcol*) [protected]

Utility:

Parameters

in	*A*	input matrix defined on finest level
in	*maxrow*	initialization of vector containing maximum values of all matrix rows (it will be built in the method)
in	*maxcol*	initialization of vector containing maximum values of all matrix columns (it will be built in the method)

Definition at line 69 of file setup.cpp.

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

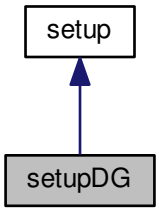
- laura/AMG_Methods/include/setup.h
- laura/AMG_Methods/src/setup.cpp

5.10 setupDG Class Reference

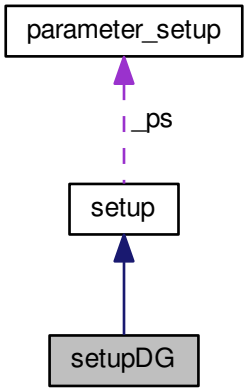
Class inherited from setup. This class defines the construction of coarser matrices and interpolation operators for matrices stemming from discontinous Galerkin discretization extending the algorithm for conforming Galerkin matrices.

`#include <setupDG.h>`

Inheritance diagram for setupDG:



Collaboration diagram for setupDG:



Public Member Functions

- setupDG ()=default

  Constructor (defaulted)
- setupDG (const SpMat &A, const parameter_setup &p)

  Constructor.
- ~setupDG ()

  Destructor (defaulted)

Private Member Functions

- void aggregation_DG (vector< sets > &B)

  Aggregation.
- void unsmoothed_interpolation (SpMat &I, vector< sets > &B)

  Unsmoothed interpolation formula.
- void GS_orth_interpolation (SpMat &I)

  Gram-Schmidt orthonormalization applied to the interpolation formula.
- void smoothed_interpolation (SpMat &I)

  Smoothing step applied to the interpolation formula.
- void DG_setup ()

  Construction of coarser matrices and interpolation operators for matrix stemming from discontinuous Galerkin discretization.
- int find_set (vector< sets > &B, const int &k)

  Find the aggregate set containing a given value.
- void maxrow_pos (const SpMat &A, vector< int > &pos)

  Utility:

Additional Inherited Members

5.10.1 Detailed Description

Class inherited from setup. This class defines the construction of coarser matrices and interpolation operators for matrices stemming from discontinous Galerkin discretization extending the algorithm for conforming Galerkin matrices.

Definition at line 25 of file setupDG.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 setupDG (const SpMat & *A*, const parameter_setup & *p*)

Constructor.

Parameters

in	*A*	input matrix defined on finest level
in	*p*	parameters of setup

Definition at line 14 of file setupDG.cpp.

5.10.3 Member Function Documentation

5.10.3.1 void aggregation_DG (vector< sets > & *B*) [private]

Aggregation.

Parameters

in	*B*	initialization of vector of aggregate sets (it will be built in the method)

Definition at line 50 of file setupDG.cpp.

5.10.3.2 void unsmoothed_interpolation (SpMat & *I*, vector< sets > & *B*) [private]

Unsmoothed interpolation formula.

Parameters

in	*I*	initialization of interpolation operator (it will be built in the method)
in	*B*	vector of aggregate sets

Definition at line 125 of file setupDG.cpp.

5.10.3.3 void GS_orth_interpolation (SpMat & *I*) [private]

Gram-Schmidt orthonormalization applied to the interpolation formula.

Parameters

in	*I*	interpolation operator

Definition at line 137 of file setupDG.cpp.

5.10.3.4 void smoothed_interpolation (SpMat & *I*) [private]

Smoothing step applied to the interpolation formula.

Parameters

in	*I*	interpolation operator

Definition at line 146 of file setupDG.cpp.

5.10.3.5 int find_set (vector< sets > & *B*, const int & *k*) [private]

Find the aggregate set containing a given value.

Parameters

in	*B*	vector of aggregate sets
in	*k*	value to be found
out	*i*	index of set containing k, if it is not found then i=-1

Definition at line 157 of file setupDG.cpp.

5.10.3.6 void maxrow_pos (const SpMat & *A*, vector< int > & *pos*) [private]

Utility:

Parameters

in	*A*	input matrix defined on finest level
in	*pos*	initialization of vector containing position of all maximum values of all matrix rows except for diagonal values (it will be built in the method)

Definition at line 169 of file setupDG.cpp.

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

- laura/AMG_Methods/include/setupDG.h
- laura/AMG_Methods/src/setupDG.cpp

5.11 GetPot::variable Struct Reference

Public Member Functions

- **variable** (const variable &)
- **variable** (const char *Name, const char *Value, const char *FieldSeparator)
- variable & **operator=** (const variable &That)
- void **take** (const char *Value, const char *FieldSeparator)
- const std::string * **get_element** (unsigned Idx) const

Public Attributes

- std::string **name**
- STRING_VECTOR **value**
- std::string **original**

5.11.1 Detailed Description

Definition at line 212 of file GetPot.h.

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

- laura/AMG_Methods/include/GetPot.h

Chapter 6

File Documentation

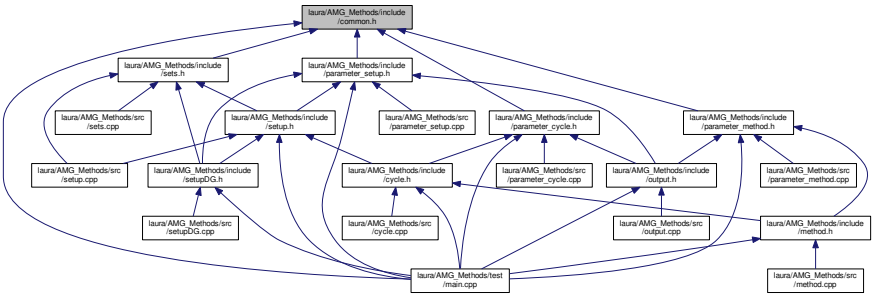
6.1 laura/AMG_Methods/include/common.h File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include <Eigen/Sparse>
#include <Eigen/SparseCore>
#include <Eigen/IterativeLinearSolvers>
#include <unsupported/Eigen/SparseExtra>
#include <iostream>
#include <fstream>
#include "stdlib.h"
#include <vector>
#include <algorithm>
#include <iterator>
#include <cmath>
```

Include dependency graph for common.h:

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



Typedefs

- using Real = double

  Typedef for real numbers.
- typedef SparseMatrix< Real > SpMat

  Typedef for sparse real-valued matrices.
- typedef SparseVector< Real > SpVec

  Typedef for sparse real-valued vectors.
- typedef SparseVector< int > SpCount

  Typedef for sparse int-valued vectors.
- using Vec = Matrix< Real, Dynamic, 1 >

  Typedef for real-valued vectors.
- typedef Triplet< Real > Trip

  Typedef for triplet, it is used to build sparse real-valued matrices.

6.1.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

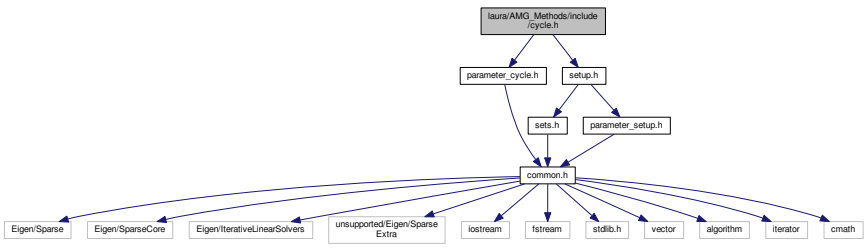
This file is part of project "AMG Methods".

6.2 laura/AMG_Methods/include/cycle.h File Reference

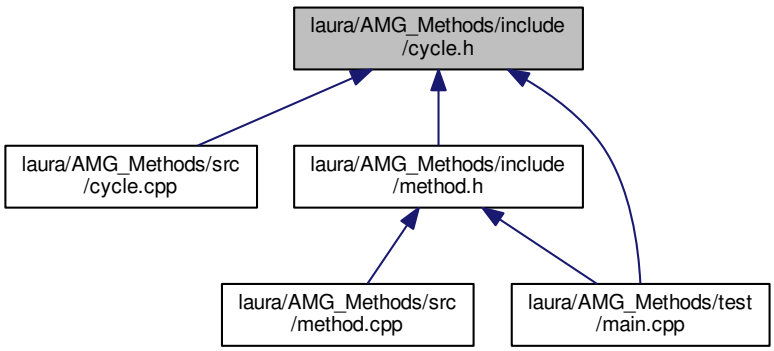
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "parameter_cycle.h"
#include "setup.h"
```

Include dependency graph for cycle.h:



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



Classes

- class cycle

  This class defines one iteration of mu-cycle.

6.2.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

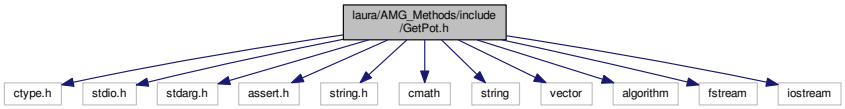
This file is part of project "AMG Methods".

6.3 laura/AMG_Methods/include/GetPot.h File Reference

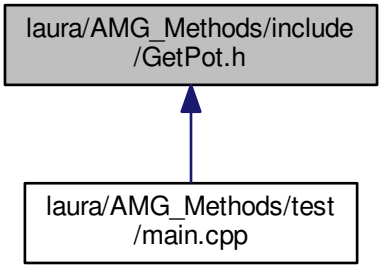
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include <ctype.h>
#include <stdio.h>
#include <stdarg.h>
#include <assert.h>
#include <string.h>
#include <cmath>
#include <string>
#include <vector>
#include <algorithm>
#include <fstream>
#include <iostream>
```

Include dependency graph for GetPot.h:



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



Classes

- class GetPot

  This class read input values from files (library GetPot http://getpot.sourceforge.net*).*
- struct GetPot::variable

Macros

- #define **victorate**(TYPE, VARIABLE, ITERATOR)

Typedefs

- typedef std::vector< std::string > **STRING_VECTOR**

6.3.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

This file is part of project "AMG Methods".

6.3.2 Macro Definition Documentation

6.3.2.1 #define victorate(*TYPE, VARIABLE, ITERATOR*)

Value:

```
std::vector<TYPE>::const_iterator ITERATOR = (VARIABLE).begin(); \
  for(; (ITERATOR) != (VARIABLE).end(); (ITERATOR)++)
```

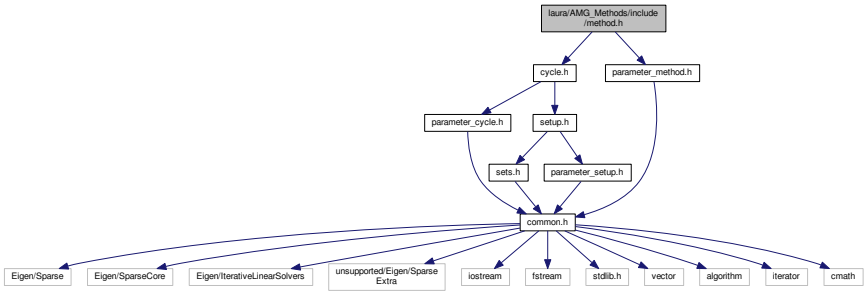
Definition at line 71 of file GetPot.h.

6.4 laura/AMG_Methods/include/method.h File Reference

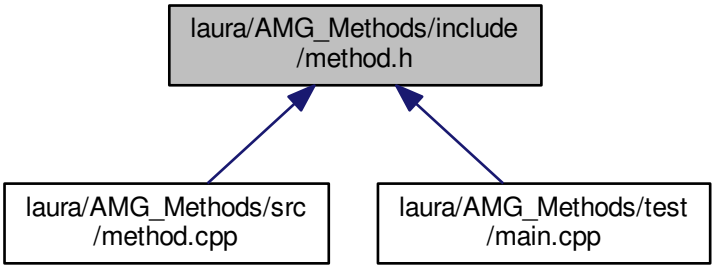
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "cycle.h"
#include "parameter_method.h"
```

Include dependency graph for method.h:



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



Classes

- class method

  This class defines AMG methods: AMG stand-alone or PCG preconditioned conjugate gradient.

6.4.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

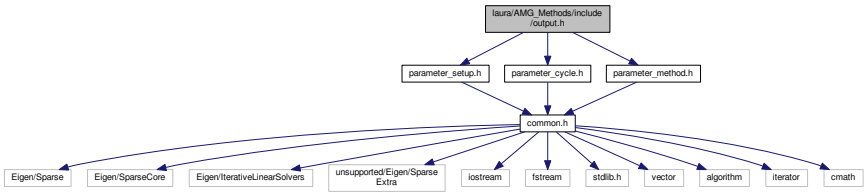
This file is part of project "AMG Methods".

6.5 laura/AMG_Methods/include/output.h File Reference

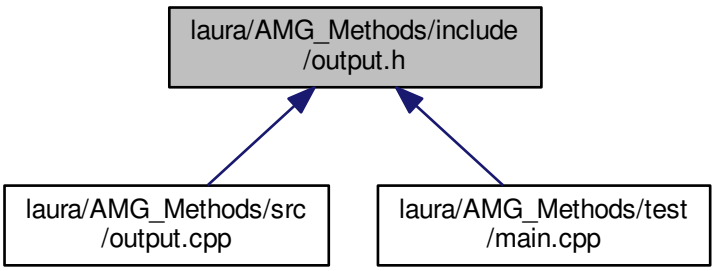
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "parameter_setup.h"
#include "parameter_cycle.h"
#include "parameter_method.h"
```

Include dependency graph for output.h:



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



Classes

- class output

  This class contains the printing and saving tools.

6.5.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

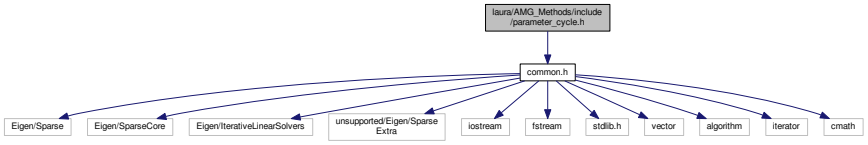
This file is part of project "AMG Methods".

6.6 laura/AMG_Methods/include/parameter_cycle.h File Reference

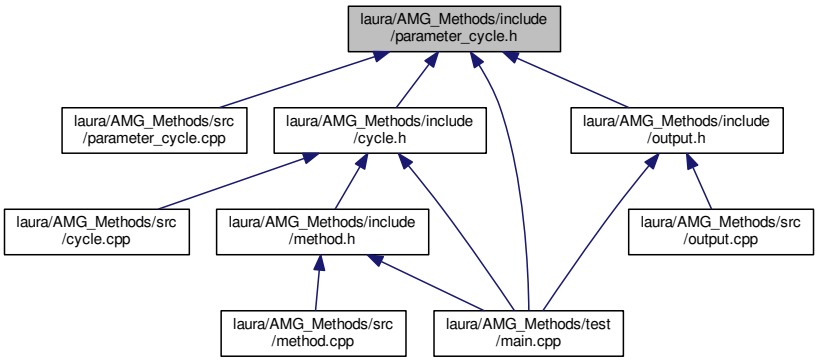
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "common.h"
```

Include dependency graph for parameter_cycle.h:



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



Classes

- class parameter_cycle

  This class contains cycle parameters.

6.6.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

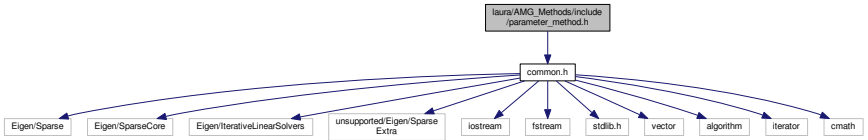
This file is part of project "AMG Methods".

6.7 laura/AMG_Methods/include/parameter_method.h File Reference

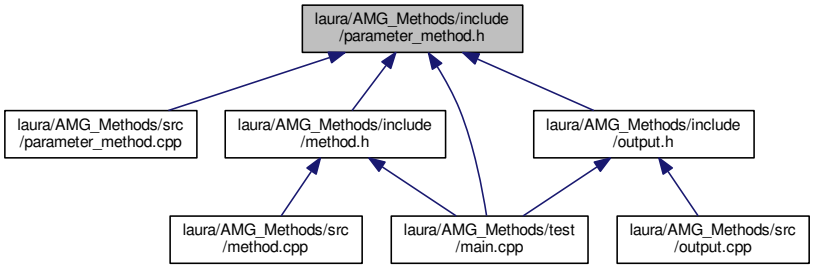
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "common.h"
```

Include dependency graph for parameter_method.h:



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



Classes

- class parameter_method

  This class contains method parameters.

6.7.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

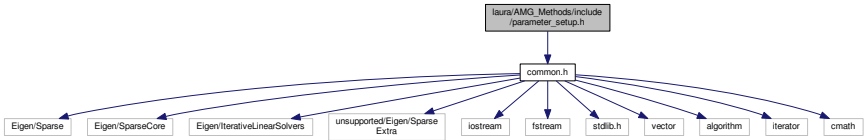
This file is part of project "AMG Methods".

6.8 laura/AMG_Methods/include/parameter_setup.h File Reference

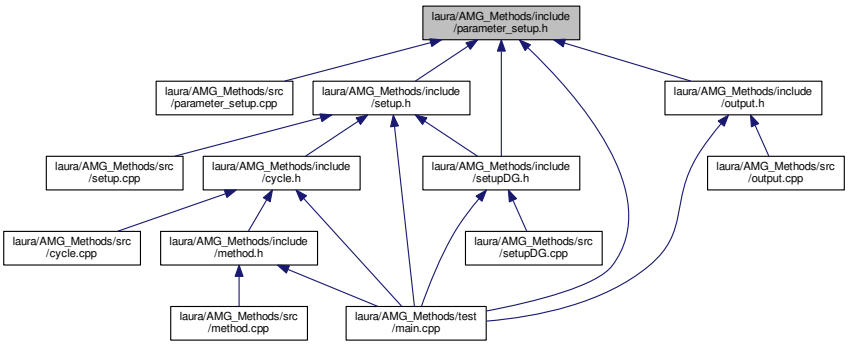
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "common.h"
```

Include dependency graph for parameter_setup.h:



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



Classes

- class parameter_setup

  This class contains setup parameters.

6.8.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

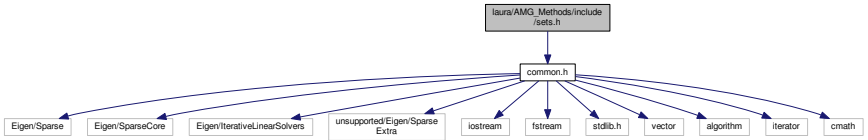
This file is part of project "AMG Methods".

6.9 laura/AMG_Methods/include/sets.h File Reference

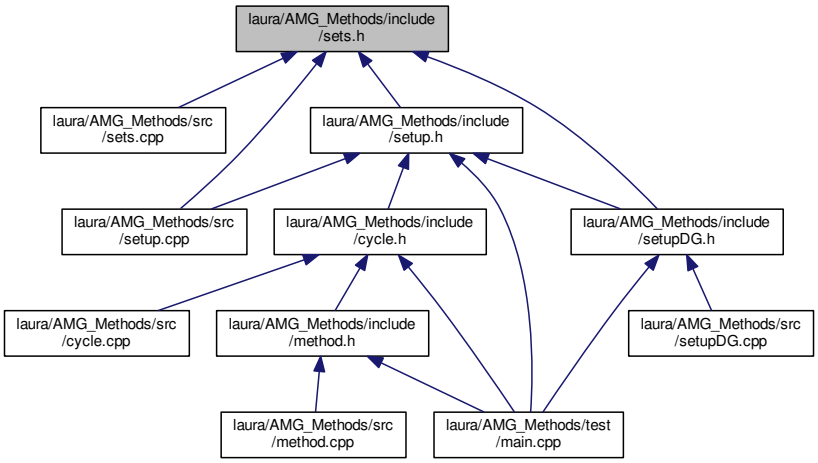
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "common.h"
```

Include dependency graph for sets.h:



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



Classes

- class sets

  This class performs some properties and utilities of mathematical sets.

6.9.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

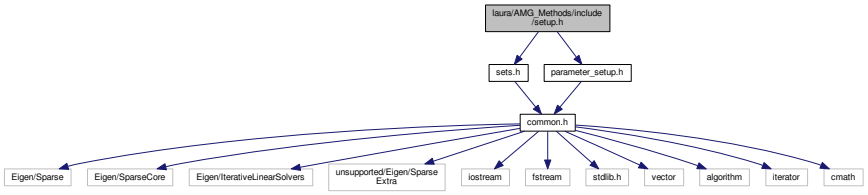
This file is part of project "AMG Methods".

6.10 laura/AMG_Methods/include/setup.h File Reference

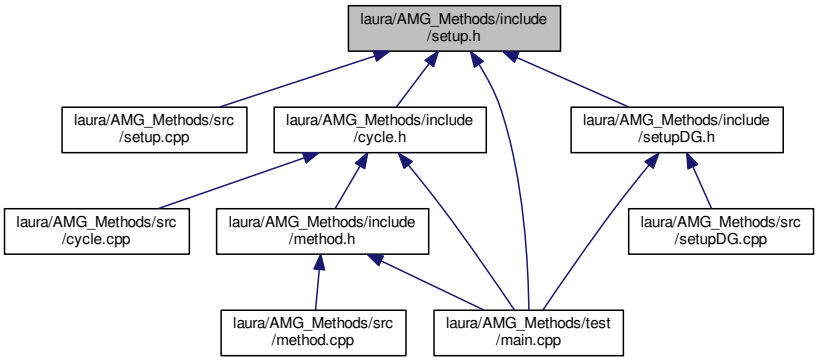
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "sets.h"
#include "parameter_setup.h"
```

Include dependency graph for setup.h:



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



Classes

- class setup

  This class defines the construction of coarser matrices and interpolation operators, in particular for matrices stemming from conforming Galerkin discretization.

6.10.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

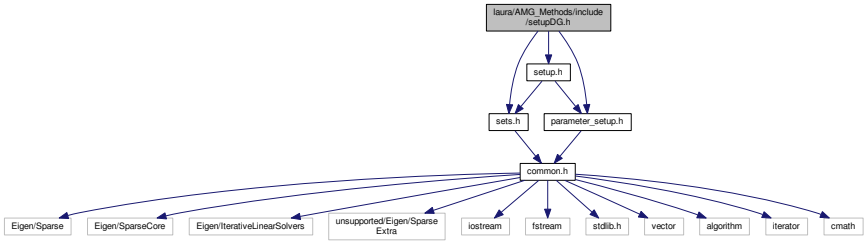
This file is part of project "AMG Methods".

6.11 laura/AMG_Methods/include/setupDG.h File Reference

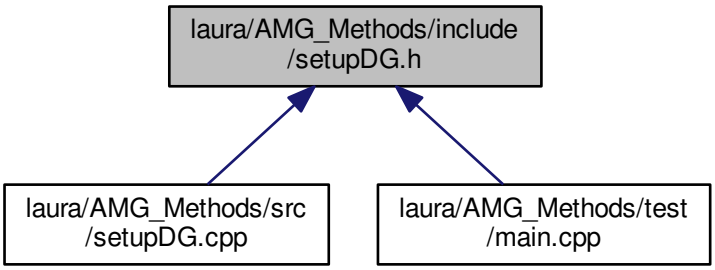
AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "sets.h"
#include "setup.h"
#include "parameter_setup.h"
```

Include dependency graph for setupDG.h:



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



Classes

- class setupDG

  Class inherited from setup. This class defines the construction of coarser matrices and interpolation operators for matrices stemming from discontinous Galerkin discretization extending the algorithm for conforming Galerkin matrices.

6.11.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

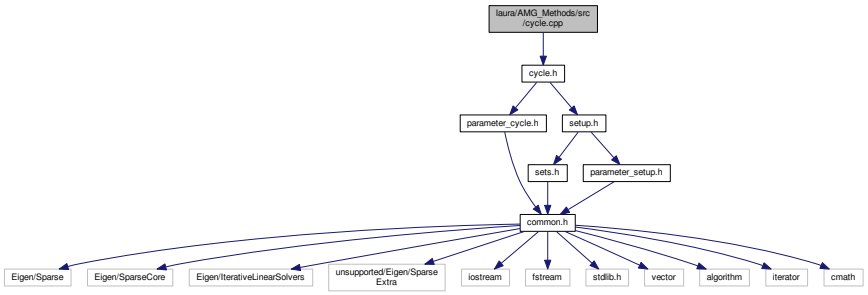
This file is part of project "AMG Methods".

6.12 laura/AMG_Methods/src/cycle.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "cycle.h"
```

Include dependency graph for cycle.cpp:



6.12.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

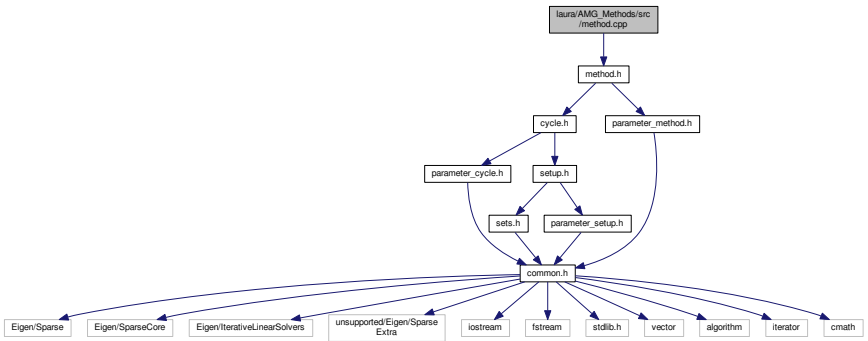
This file is part of project "AMG Methods".

6.13 laura/AMG_Methods/src/method.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "method.h"
```

Include dependency graph for method.cpp:



6.13.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

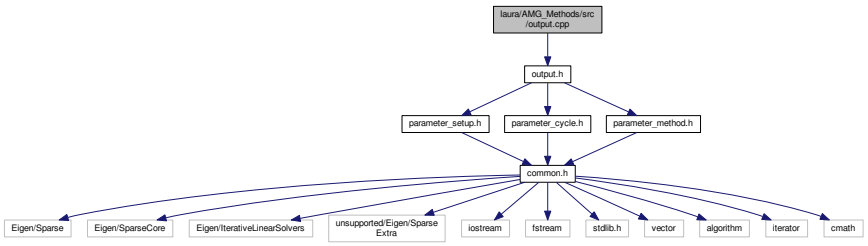
This file is part of project "AMG Methods".

6.14 laura/AMG_Methods/src/output.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "output.h"
```

Include dependency graph for output.cpp:



6.14.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

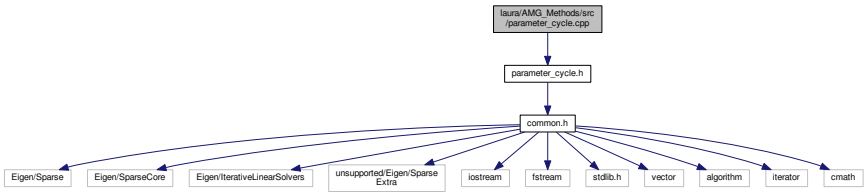
2017

This file is part of project "AMG Methods".

6.15 laura/AMG_Methods/src/parameter_cycle.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "parameter_cycle.h"
```
Include dependency graph for parameter_cycle.cpp:



6.15.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

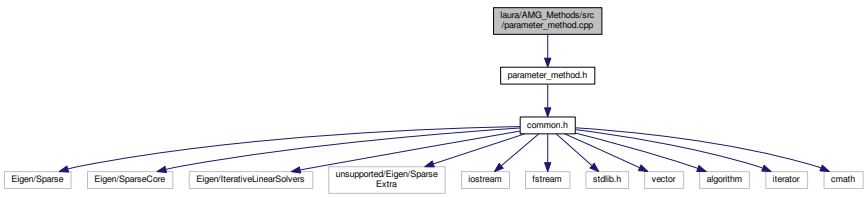
2017

This file is part of project "AMG Methods".

6.16 laura/AMG_Methods/src/parameter_method.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "parameter_method.h"
```
Include dependency graph for parameter_method.cpp:



6.16.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

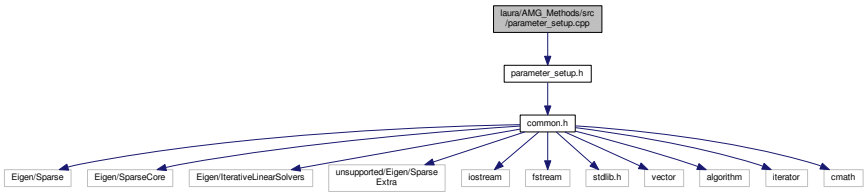
This file is part of project "AMG Methods".

6.17 laura/AMG_Methods/src/parameter_setup.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "parameter_setup.h"
```

Include dependency graph for parameter_setup.cpp:



6.17.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

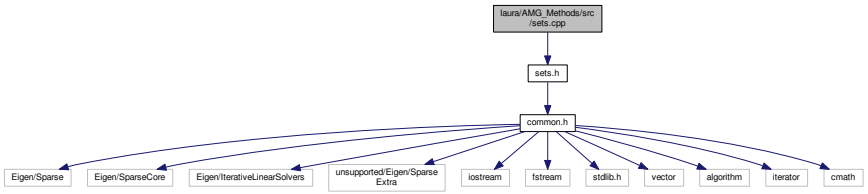
2017

This file is part of project "AMG Methods".

6.18 laura/AMG_Methods/src/sets.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "sets.h"
```
Include dependency graph for sets.cpp:



6.18.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

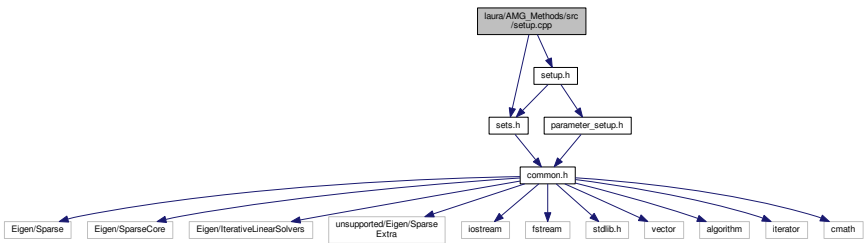
2017

This file is part of project "AMG Methods".

6.19 laura/AMG_Methods/src/setup.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "setup.h"
#include "sets.h"
```
Include dependency graph for setup.cpp:



6.19.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

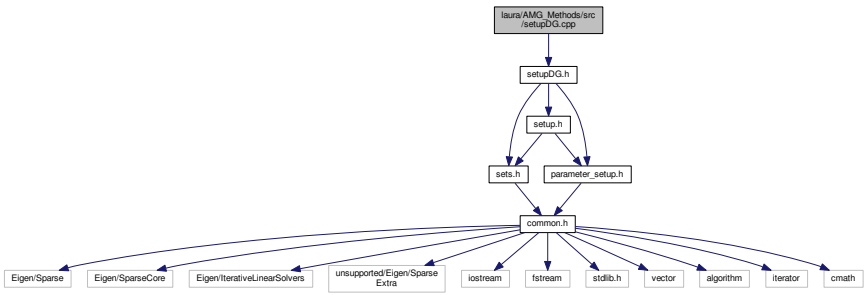
This file is part of project "AMG Methods".

6.20 laura/AMG_Methods/src/setupDG.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "setupDG.h"
```

Include dependency graph for setupDG.cpp:



6.20.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

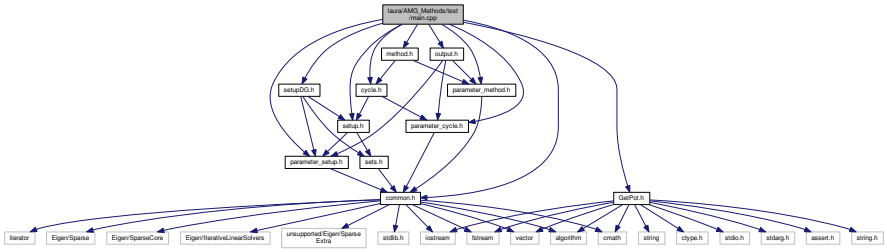
This file is part of project "AMG Methods".

6.21 laura/AMG_Methods/test/main.cpp File Reference

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

```
#include "common.h"
#include "GetPot.h"
#include "parameter_setup.h"
#include "parameter_cycle.h"
#include "parameter_method.h"
#include "setup.h"
#include "setupDG.h"
#include "cycle.h"
#include "method.h"
#include "output.h"
```

Include dependency graph for main.cpp:



Functions

- int main (const int argc, char ∗argv[ ])

  *The **main** function.*

6.21.1 Detailed Description

AMG methods for conforming and discontinuous Galerkin finite element discretizations of the Poisson problem.

Author

Laura Melas laura.melas@mail.polimi.it

Date

2017

This file is part of project "AMG Methods".

6.21.2 Function Documentation

6.21.2.1 int main (const int *argc,* char * *argv[ ]*)

The **main** function.

Read input/output and files parameters.

Read setup parameters.

Read cycle parameters.

Read method parameters.

Instantiate setup.

Instantiate cycle.

Instantiate method.

Apply AMG.

Print output.

Save solution.

Definition at line 27 of file main.cpp.

Index