deal.II ifpack Interface for BoomerAMG use as a Solver or Preconditioner

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 TrilinosWrappers::BoomerAMG_PreconditionedSolver Class Reference

This class serves as an interface to ifpack for using a hypre solver with a BoomerAMG preconditioner.

Public Member Functions

- BoomerAMG_PreconditionedSolver (BoomerAMGParameters &BoomerAMG_precond_parameters, ifpack↩ SolverParameters &solver_parameters)

  Constructor.
- void solve (LA::SparseMatrix &system_matrix, LA::Vector &right_hand_side, LA::Vector &solution)

  Solver function.

3.1.1 Detailed Description

This class serves as an interface to ifpack for using a hypre solver with a BoomerAMG preconditioner.

Author

Joshua Hanophy, 2019

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BoomerAMG_PreconditionedSolver()

```
TrilinosWrappers::BoomerAMG_PreconditionedSolver::BoomerAMG_PreconditionedSolver (
            BoomerAMGParameters & BoomerAMG_precond_parameters,
            ifpackSolverParameters & solver_parameters )  [inline]
```

Constructor.

Parameters

BoomerAMG_precond_parameters	is the instance of BoomerAMGParameters hanlding the BoomerAMG parameters
solver_parameters	is the instance of ifpackSolverParameters hanlding the solver parameters

3.1.3 Member Function Documentation

3.1.3.1 solve()

```
void TrilinosWrappers::BoomerAMG_PreconditionedSolver::solve (
            LA::SparseMatrix & system_matrix,
            LA::Vector & right_hand_side,
            LA::Vector & solution )
```

Solver function.

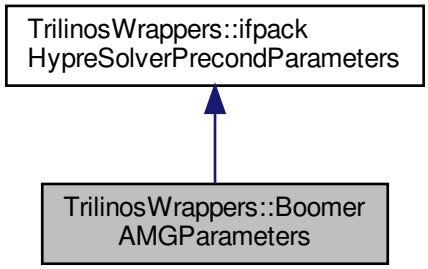
Parameters

system_matrix	is the system matrix
right_hand_side	it the right hand side for the system
solution	is the solution vector into which the solution will be written

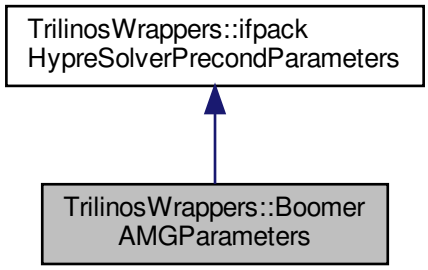
3.2 TrilinosWrappers::BoomerAMGParameters Class Reference

Class meant to handle BoomerAMG solver or preconditioner parameters.

Inheritance diagram for TrilinosWrappers::BoomerAMGParameters:



Collaboration diagram for TrilinosWrappers::BoomerAMGParameters:



Public Types

- enum AMG_type { CLASSICAL_AMG, AIR_AMG, NONE }
  Enum storing possible default parameter configurations.

Public Member Functions

- BoomerAMGParameters (AMG_type config_selection, Hypre_Chooser solver_preconditioner_selection)
  Constructor.

Additional Inherited Members

3.2.1 Detailed Description

Class meant to handle BoomerAMG solver or preconditioner parameters.

This class adds little functionality to its base class, but includes a comprehensive list of default parameters that may be of interest for BoomerAMG when used as either a solver or a preconditioner. The constructor for this class is of primary interest and sets a number of parameters.

The default parameters are:

String Name	Description

interp_type	The interp_type integer variable sets the interpolation type. Interpolation types, taken from the hypre documentation, are: • 0: classical modified interpolation • 1: LS interpolation (for use with GSMG) • 2: classical modified interpolation for hyperbolic PDEs • 3: direct interpolation (with separation of weights) • 4: multipass interpolation • 5: multipass interpolation (with separation of weights) • 7: extended+i interpolation • 8: standard interpolation • 9: standard interpolation (with separation of weights) • 10: classical block interpolation (for use with nodal systems version only) • 11: classical block interpolation (for use with nodal systems version only) with diagonalized diagonal blocks • 12: FF interpolation • 13: FF1 interpolation • 14: extended interpolation • 100: Pointwise interpolation (intended for use with AIR)
pre_post_relax	The prerelax string specifies the points, order, and relaxation steps for prerelaxation. The options are "A", "F", or "C" where A is relaxation over all points, F is relaxation over the F-points, and C is relaxation over the C-points. Multiple characters specify multiple relaxation steps and the order matters. For example, "AA" specifies two relaxation steps of all points. The postrelax string specifies the points, order, and relaxation steps for postrelaxation. The options are "A", "F", or "C" where A is relaxation over all points, F is relaxation over the F-points, and C is relaxation over the C-points. Multiple characters specify multiple relaxation steps and the order matters. For example, "FFFC" specifies three post relaxations over F-points followed by a relexation over C-points.

relax_type	The relax_type integer variable sets the relaxation type. Relaxation types, taken from the hypre documentation, are: • 0: Jacobi • 1: Gauss-Seidel, sequential (very slow!) • 2: Gauss-Seidel, interior points in parallel, boundary sequential (slow!) • 3: hybrid Gauss-Seidel or SOR, forward solve • 4: hybrid Gauss-Seidel or SOR, backward solve • 5: hybrid chaotic Gauss-Seidel (works only with OpenMP) • 6: hybrid symmetric Gauss-Seidel or SSOR • 8: ℓ_1 Gauss-Seidel, forward solve • 9: Gaussian elimination (only on coarsest level) • 13: ℓ_1 Gauss-Seidel, forward solve • 14: ℓ_1 Gauss-Seidel, backward solve • 15: CG (warning - not a fixed smoother - may require FGMRES) • 16: Chebyshev • 17: FCF-Jacobi • 18: ℓ_1-scaled jacobi
coarsen_type	The coarsen_type integer variable sets the coarsening algorithm. Coarsening algorithm options, taken from the hypre documentation, are: • 0: CLJP-coarsening (a parallel coarsening algorithm using independent sets. • 3: classical Ruge-Stueben coarsening on each processor, followed by a third pass, which adds coarse points on the boundaries • 6: Falgout coarsening (uses 1 first, followed by CLJP using the interior coarse points generated by 1 as its first independent set) • 8: PMIS-coarsening (a parallel coarsening algorithm using independent sets, generating lower complexities than CLJP, might also lead to slower convergence) • 10: HMIS-coarsening (uses one pass Ruge-Stueben on each processor independently, followed by PMIS using the interior C-points generated as its first independent set) • 21: CGC coarsening by M. Griebel, B. Metsch and A. Schweitzer • 22: CGC-E coarsening by M. Griebel, B. Metsch and A.Schweitzer
max_levels	The max_levels integer specifies the maximum number of AMG that hypre will be allowed to use

cycle_type	The cycle_type integer variable sets the cycle type. Cycle types available, taken from the hypre documentation, are: • 0: F-cycle type • 1: V-cycle type • 2: W-cycle type
sabs_flag	sabs_flag sets whether the classical strength of connection test based on testing the negative of matrix coefficient or if the absolute value is tested. If set to 0, the negative coefficient values are tested, if set to 1, the absolute values of matrix coefficients are tested.
distance_R	The distance_R double variable sets whether Approximate Ideal Restriction (AIR) multigrid or classical multigrid is used. • 0.0: Use classical AMG, not AIR • 1.0: Use AIR, Distance-1 LAIR is used to compute R • 2.0: Use AIR, Distance-2 LAIR is used to compute R • 3.0: Use AIR, degree 0 Neumann expansion is used to compute R • 4.0: Use AIR, degree 1 Neumann expansion is used to compute R • 5.0: Use AIR, degree 2 Neumann expansion is used to compute R

Author

Joshua Hanophy, Ben Southworth 2019

3.2.2 Member Enumeration Documentation

3.2.2.1 AMG_type

`enum TrilinosWrappers::BoomerAMGParameters::AMG_type`

Enum storing possible default parameter configurations.

Generally, different sets of parameters may be of interest for different AMG solvers. The AMG_type enum determines which parameters are set when an instance of BoomerAMGParameters is constructed.

Enumerator

CLASSICAL_AMG	Load default parameters consistent with Classical AMG. CLASSICAL_AMG
AIR_AMG	Load default parameters consistent with AIR. AIR_AMG
NONE	Create an empty parameter map. NONE

3.2.3 Constructor & Destructor Documentation

3.2.3.1 BoomerAMGParameters()

```
TrilinosWrappers::BoomerAMGParameters::BoomerAMGParameters (
            AMG_type config_selection,
            Hypre_Chooser solver_preconditioner_selection )
```

Constructor.

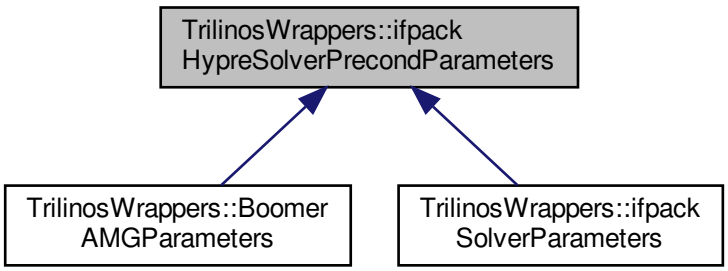
Parameters

config_selection	is the AMG_type specifying what default parameters should be used
solver_preconditioner_selection	is either Hypre_Chooser:Solver or Hypre_Chooser:Preconditioner and specifies whether the parameter object will be used of a solver or a preconditioner.

3.3 TrilinosWrappers::ifpackHypreSolverPrecondParameters Class Reference

This class is meatn to handle parameters that are used by hypre solvers and preconditioners.

Inheritance diagram for TrilinosWrappers::ifpackHypreSolverPrecondParameters:



Classes

- struct parameter_data

  This struct holds the data required for a single parameter.

Public Types

- typedef boost::variant< int(∗)(HYPRE_Solver, int), int(∗)(HYPRE_Solver, double), int(∗)(HYPRE_Solver, double, int), int(∗)(HYPRE_Solver, int, int), int(∗)(HYPRE_Solver, int ∗), int(∗)(HYPRE_Solver, double ∗), int(∗)(HYPRE_Solver, int ∗∗), std::nullptr_t > hypre_function_variant

  This type is meant to be used for a pointer to a hypre set function.
- typedef boost::variant< int, double, int ∗, int ∗∗, std::pair< double, int >, std::pair< int, int >, std::pair< std::string, std::string > > param_value_variant

  This type is meant to be used for a parameter value.

Public Member Functions

- ifpackHypreSolverPrecondParameters (Hypre_Chooser solver_preconditioner_selection)

  Constructor.
- void set_parameter_value (std::string name, param_value_variant value)

  This function can be used to change the value of a parameter in the parameters map that already exists.
- void add_parameter (std::string name, parameter_data param_data)

  This function can be used to add a new parameter to the.
- void remove_parameter (std::string name)

  Function to remove a parameter from the parameters parameter map.
- template<typename return_type >
  void return_parameter_value (std::string name)

  Function to return the value of a parameter.
- void set_parameters (Ifpack_Hypre &Ifpack_obj)

  This function is to be used by the solver or preconditioner class to set the parameter values.

Protected Attributes

- std::map< std::string, parameter_data > parameters

  The parameters map stores parameters as a string name key and then a parameter_data instance value.

3.3.1 Detailed Description

This class is meatn to handle parameters that are used by hypre solvers and preconditioners.

It stores parameter values and also interfaces with an ifpack_Hypre object to actually set the parameter values.

Author

Joshua Hanophy 2019

3.3.2 Member Typedef Documentation

3.3.2.1 hypre_function_variant

```
typedef boost::variant<int (*)(HYPRE_Solver, int),int (*)(HYPRE_Solver, double),int (*)(HYPR↩
E_Solver,double, int), int (*)(HYPRE_Solver, int, int),int (*)(HYPRE_Solver, int*),int (*)(H↩
YPRE_Solver, double*), int (*)(HYPRE_Solver, int**),std::nullptr_t> TrilinosWrappers::ifpack↩
HypreSolverPrecondParameters::hypre_function_variant
```

This type is meant to be used for a pointer to a hypre set function.

This is simply an alias for a boost variant type. Note the types in the boos variant contianer are those function pointer types supported by this interface.

3.3.2.2 param_value_variant

```
typedef boost::variant<int,double,int*,int**,std::pair<double,int>, std::pair<int, int>,
std::pair<std::string,std::string> > TrilinosWrappers::ifpackHypreSolverPrecondParameters↩
::param_value_variant
```

This type is meant to be used for a parameter value.

This is simply an alias for a boost variant type.

3.3.3 Constructor & Destructor Documentation

3.3.3.1 ifpackHypreSolverPrecondParameters()

```
TrilinosWrappers::ifpackHypreSolverPrecondParameters::ifpackHypreSolverPrecondParameters (
            Hypre_Chooser solver_preconditioner_selection )  [inline]
```

Constructor.

Parameters

solver_preconditioner_selection	is either a Hypre_Chooser enum::Solver or Hypre_Chooser enum::Solver preconditioner and specifies whether the instance will handle paramets for a solver or a preconditioner.

3.3.4 Member Function Documentation

3.3.4.1 add_parameter()

```
void TrilinosWrappers::ifpackHypreSolverPrecondParameters::add_parameter (
            std::string name,
            parameter_data param_data )
```

This function can be used to add a new parameter to the.

Parameters

name	is the string parameter name. Note that the parameter name should not already exist. To update the value or a parameter that already exists, use the set_parameter_value function
param_data	is an instance of the parameter_data struct which contains the parameter value and a pointer to the proper set function

3.3.4.2 remove_parameter()

```
void TrilinosWrappers::ifpackHypreSolverPrecondParameters::remove_parameter (
            std::string name )
```

Function to remove a parameter from the parameters parameter map.

Parameters

name	is the string parameter name to remove.

3.3.4.3 return_parameter_value()

```
template<typename return_type >
void TrilinosWrappers::ifpackHypreSolverPrecondParameters::return_parameter_value (
            std::string name )
```

Function to return the value of a parameter.

Parameters

name	is the string parameter name of the parameter whose value is to be returned

3.3.4.4 set_parameter_value()

```
void TrilinosWrappers::ifpackHypreSolverPrecondParameters::set_parameter_value (
            std::string name,
            param_value_variant value )
```

This function can be used to change the value of a parameter in the parameters map that already exists.

Use the add_parameter function if the parameter does not already exist

Parameters

name	is the string parameter name. Note that the parameter name should already exist in the parameters parameter map. Use the add_parameter function to add a new parameters.
value	This is the value to assign the parameter found at parameters[name]

3.3.4.5 set_parameters()

```
void TrilinosWrappers::ifpackHypreSolverPrecondParameters::set_parameters (
            Ifpack_Hypre & Ifpack_obj )
```

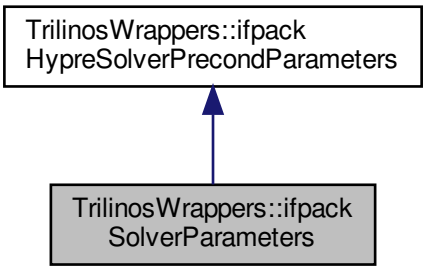
This function is to be used by the solver or preconditioner class to set the parameter values.

Note that all parameters contained in the parameters map will be set.

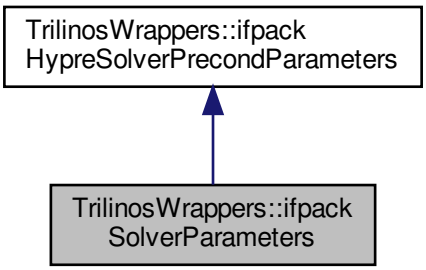
3.4 TrilinosWrappers::ifpackSolverParameters Class Reference

This class adds minimal functionality to its base class.

Inheritance diagram for TrilinosWrappers::ifpackSolverParameters:



Collaboration diagram for TrilinosWrappers::ifpackSolverParameters:



Public Member Functions

- ifpackSolverParameters (Hypre_Solver solver_selection=Hypre_Solver::PCG)
  Constructor.

Public Attributes

- Hypre_Solver **solver_selection**

Additional Inherited Members

3.4.1 Detailed Description

This class adds minimal functionality to its base class.

It is meant to be used to handle parameters for a hypre solver other than BoomerAMG. In particular, it is used by BoomerAMG_PreconditionedSolver to handle the solver parameters.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ifpackSolverParameters()

```
TrilinosWrappers::ifpackSolverParameters::ifpackSolverParameters (
            Hypre_Solver solver_selection = Hypre_Solver::PCG )  [inline]
```

Constructor.

Parameters

solver_selection	specifies the solver type to be used.

3.5 TrilinosWrappers::ifpackHypreSolverPrecondParameters::parameter_data Struct Reference

This struct holds the data required for a single parameter.

Public Member Functions

- parameter_data (param_value_variant value, hypre_function_variant hypre_function)
  Constructor.
- parameter_data (param_value_variant value, std::function< void(const Hypre_Chooser, const parameter_data &, Ifpack_Hypre &)> set_function)
  Constructor.

Public Attributes

- param_value_variant value
  value stores the value of the parameter
- hypre_function_variant hypre_function =nullptr
  hypre_function stores a pointer to the hypre set function to be used to set the parameter value.
- std::function< void(const Hypre_Chooser, const parameter_data &, Ifpack_Hypre &)> set_function =nullptr
  set_function stores a pointer to a custom set function.

3.5.1 Detailed Description

This struct holds the data required for a single parameter.

The required data is a parameter value and a set function. There are two possibilities for a set function. A pointer to a hypre set function defined in the hypre library can used and will be stored as hypre_function. Or if a custom set function is desired in place of directly using the predfined hypre set functions, a pointer to a custom set function is stored as set_function. Note that either set_function or hypre_function shoudl be a nullptr.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 parameter_data() [1/2]

```
TrilinosWrappers::ifpackHypreSolverPrecondParameters::parameter_data::parameter_data (
            param_value_variant value,
            hypre_function_variant hypre_function )  [inline]
```

Constructor.

Parameters

value	is the value of the parameter
hypre_function	is a pointer to the hypre set function defined in the hypre library

3.5.2.2 parameter_data() [2/2]

```
TrilinosWrappers::ifpackHypreSolverPrecondParameters::parameter_data::parameter_data (
            param_value_variant value,
            std::function< void(const Hypre_Chooser, const parameter_data &, Ifpack_Hypre &)>
set_function )  [inline]
```

Constructor.

Parameters

value	is the value of the parameter
set_function	is a pointer to a custom set function

3.5.3 Member Data Documentation

3.5.3.1 hypre_function

```
hypre_function_variant TrilinosWrappers::ifpackHypreSolverPrecondParameters::parameter_data↩
::hypre_function =nullptr
```

hypre_function stores a pointer to the hypre set function to be used to set the parameter value.

Note is this is used then hypre_function should be equal to a nullptr

3.5.3.2 set_function

```
std::function<void(const Hypre_Chooser, const parameter_data &, Ifpack_Hypre &)> Trilinos↩
Wrappers::ifpackHypreSolverPrecondParameters::parameter_data::set_function =nullptr
```

set_function stores a pointer to a custom set function.

This can be used if simply setting a parameter value with the set functions predifined in the hypre library is not sufficient. If this is used, hypre_function should be equal to nullptr

3.6 TrilinosWrappers::SolverBoomerAMG Class Reference

This class serves as an interface to ifpack for using a BoomerAMG as a solver.

Public Member Functions

- SolverBoomerAMG (BoomerAMGParameters &SolverParameters)
  Constructor.
- void solve (LA::SparseMatrix &system_matrix, LA::Vector &right_hand_side, LA::Vector &solution)
  Solver function.

3.6.1 Detailed Description

This class serves as an interface to ifpack for using a BoomerAMG as a solver.

Author

Joshua Hanophy, 2019

3.6.2 Constructor & Destructor Documentation

3.6.2.1 SolverBoomerAMG()

```
TrilinosWrappers::SolverBoomerAMG::SolverBoomerAMG (
            BoomerAMGParameters & SolverParameters )  [inline]
```

Constructor.

Parameters

SolverParameters	is the instance of BoomerAMGParameters container the parameter that the solver will use.

3.6.3 Member Function Documentation

3.6.3.1 solve()

```
void TrilinosWrappers::SolverBoomerAMG::solve (
             LA::SparseMatrix & system_matrix,
             LA::Vector & right_hand_side,
             LA::Vector & solution )
```

Solver function.

Parameters

system_matrix	is the system matrix
right_hand_side	it the right hand side for the system
solution	is the solution vector into which the solution will be written