```
    ______ _                      _____       __               
   / ____/(_)___ ____  ____      / ___/____  / /   _____  _____
  / __/ / / __ `/ _ \/ __ \     \__ \/ __ \/ / | / / _ \/ ___/
 / /___/ / /_/ /  __/ / / /    ___/ / /_/ / /| |/ /  __/ /    
/_____/_/\__, /\___/_/ /_/    /____/\____/_/ |___/\___/_/     
        /____/                                                 
```

Eigenvalue Solver Project

Generated by Doxygen 1.9.1

Chapter 1

Test List

Member TEST_F (InputTest, ParsesValidRealMatrix)
InputTest::ParsesValidRealMatrix

Member TEST_F (InputTest, ParsesValidComplexMatrix)
InputTest::ParsesValidComplexMatrix

Member TEST_F (InputTest, ThrowsForInvalidMatrixDimensions)
InputTest::ThrowsForInvalidMatrixDimensions

Member TEST_F (InputTest, ThrowsForInvalidMethod)
InputTest::ThrowsForInvalidMethod

Member TEST_F (InputTest, ParsesSingularMatrix)
InputTest::ParsesSingularMatrix

Member TEST_F (InputTest, ThrowsForMissingMatrixData)
InputTest::ThrowsForMissingMatrixData

Member TEST_F (SolverTest, PowerMethodSolverRealMatrix)
SolverTest::PowerMethodSolverRealMatrix

Member TEST_F (SolverTest, InversePowerMethodSolverRealMatrix)
SolverTest::InversePowerMethodSolverRealMatrix

Member TEST_F (SolverTest, QRMethodSolverRealMatrix)
SolverTest::QRMethodSolverRealMatrix

Member TEST_F (SolverTest, PowerMethodSolverComplexMatrix)
SolverTest::PowerMethodSolverComplexMatrix

Member TEST_F (SolverTest, QRMethodSolverComplexMatrix)
SolverTest::QRMethodSolverComplexMatrix

Member TEST_F (SolverTest, InversePowerMethodSolverSingularRealMatrixThrows)
SolverTest::InversePowerMethodSolverSingularRealMatrixThrows

Member TEST_F (SolverTest, InversePowerMethodSolverSingularRealMatrixThrowsWithShift)
SolverTest::InversePowerMethodSolverSingularRealMatrixThrowsWithShift

Member TEST_F (SolverTest, QRConvergenceTest)
SolverTest::QRConvergenceTest

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 files with brief descriptions:

Chapter 5

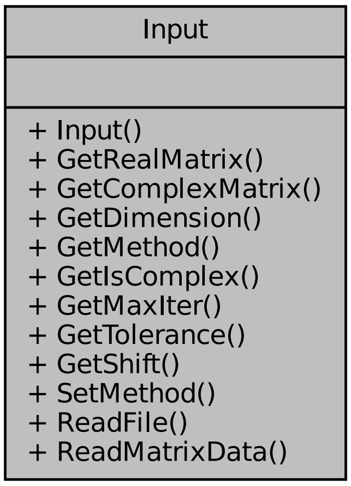
Class Documentation

5.1 Input Class Reference

Class to handle numerical input parameters and matrices from a file.

```
#include <input.hpp>
```

Collaboration diagram for Input:



Public Member Functions

- Input ()
- Eigen::MatrixXd GetRealMatrix () const

  Get the real matrix.
- Eigen::MatrixXcd GetComplexMatrix () const

Get the complex matrix.
- int GetDimension () const

  Get the dimension of the square matrix.
- std::string GetMethod () const

  Get the numerical method to be used.
- bool GetIsComplex () const

  Check whether the matrix is complex or real.
- int GetMaxIter () const

  Get the maximum number of iterations.
- double GetTolerance () const

  Get the convergence tolerance.
- double GetShift () const

  Get the shift value for the inverse power method.
- void SetMethod (std::string &method)

  Set the numerical method to be used.
- void ReadFile (const std::string &fileName)

  Read numerical parameters and matrix data from a file.
- void ReadMatrixData (std::ifstream &file, int rows, int cols)

  Read matrix elements from the input file.

5.1.1 Detailed Description

Class to handle numerical input parameters and matrices from a file.

The Input class provides methods to read configuration data and matrices from a file and store them for numerical computations. It supports both real and complex matrices.

Note

The matrix must be square, and numerical parameters such as tolerance and maximum iterations have specific constraints.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Input()

```
Input::Input ( )  [inline]
```

5.1.3 Member Function Documentation

5.1.3.1 GetComplexMatrix()

```
Eigen::MatrixXcd Input::GetComplexMatrix ( ) const
```

Get the complex matrix.

Returns

A copy of the complex matrix as an Eigen::MatrixXcd.

5.1.3.2 GetDimension()

```
int Input::GetDimension ( ) const
```

Get the dimension of the square matrix.

Returns

The dimension as an integer.

5.1.3.3 GetIsComplex()

```
bool Input::GetIsComplex ( ) const
```

Check whether the matrix is complex or real.

Returns

True if the matrix is complex, false otherwise.

5.1.3.4 GetMaxIter()

```
int Input::GetMaxIter ( ) const
```

Get the maximum number of iterations.

Returns

The maximum number of iterations as an integer.

5.1.3.5 GetMethod()

```
std::string Input::GetMethod ( ) const
```

Get the numerical method to be used.

Returns

A string representing the numerical method.

5.1.3.6 GetRealMatrix()

```
Eigen::MatrixXd Input::GetRealMatrix ( ) const
```

Get the real matrix.

Returns

A copy of the real matrix as an Eigen::MatrixXd.

5.1.3.7 GetShift()

```
double Input::GetShift ( ) const
```

Get the shift value for the inverse power method.

Returns

The shift value as a double.

5.1.3.8 GetTolerance()

```
double Input::GetTolerance ( ) const
```

Get the convergence tolerance.

Returns

The tolerance value as a double.

5.1.3.9 ReadFile()

```
void Input::ReadFile (
             const std::string & fileName )
```

Read numerical parameters and matrix data from a file.

Parses the input file to initialize the numerical parameters and read the matrix data. The file must adhere to a specific format with clearly defined sections for parameters and the matrix.

Parameters

fileName	The name of the input file to read.

Exceptions

std::runtime_error	If the file cannot be opened, contains invalid parameters, or is missing the required MATRIX_DATA section.

References ReadMatrixData().

5.1.3.10 ReadMatrixData()

```
void Input::ReadMatrixData (
            std::ifstream & file,
            int rows,
            int cols )
```

Read matrix elements from the input file.

Parses matrix data from the file after reading its dimensions. The function supports both real and complex matrices based on the `mIsComplex` flag.

Parameters

file	Reference to the input file stream.
rows	Number of rows in the matrix.
cols	Number of columns in the matrix.

Exceptions

std::runtime_error	If the data is incomplete, mismatched with dimensions, or contains invalid entries.

5.1.3.11 SetMethod()

```
void Input::SetMethod (
            std::string & method )
```

Set the numerical method to be used.

Parameters

method	A string representing the method.

Exceptions

std::runtime_error	If the method is not one of the valid options.

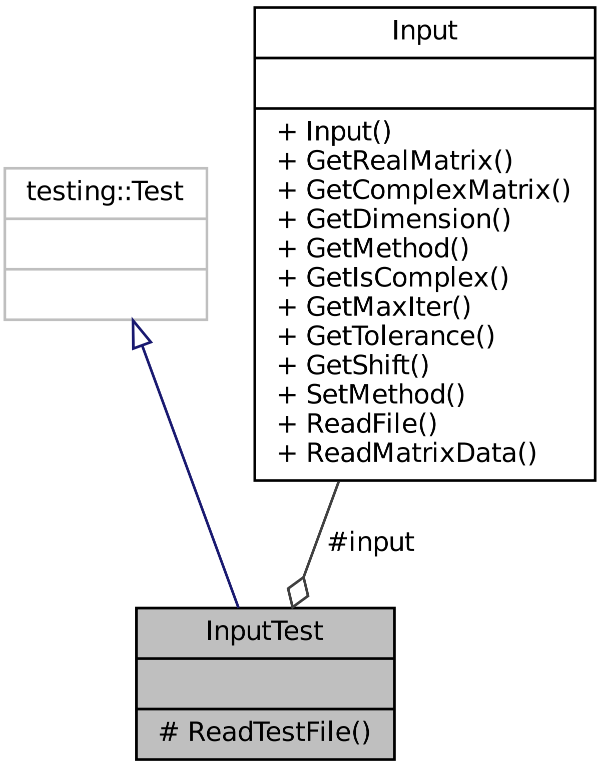
The documentation for this class was generated from the following files:

- src/input.hpp
- src/input.cpp

5.2 InputTest Class Reference

Google Test fixture for testing the `Input` class.

Collaboration diagram for InputTest:



Protected Member Functions

- void ReadTestFile (const std::string &fileName, const std::string &content)

  Helper function to read and parse a temporary test file.

Protected Attributes

- Input input

  Instance of `Input` for testing.

5.2.1 Detailed Description

Google Test fixture for testing the `Input` class.

Provides helper methods for reading test files and cleaning up temporary data.

5.2.2 Member Function Documentation

5.2.2.1 ReadTestFile()

```
void InputTest::ReadTestFile (
            const std::string & fileName,
            const std::string & content )  [inline], [protected]
```

Helper function to read and parse a temporary test file.

Parameters

fileName	Name of the file to read.
content	Content to write to the file.

References CleanUpTempFile(), input, Input::ReadFile(), and WriteToTempFile().

5.2.3 Member Data Documentation

5.2.3.1 input

```
Input InputTest::input  [protected]
```

Instance of `Input` for testing.

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

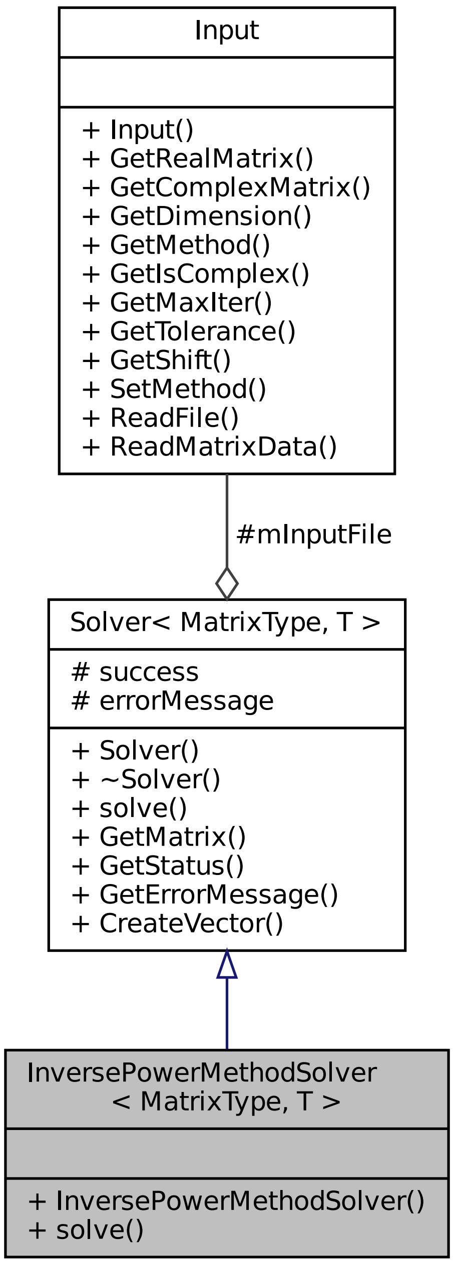
- test/test.cpp

5.3 InversePowerMethodSolver< MatrixType, T > Class Template Reference

Solver for computing an eigenvalue close to a given shift using the Inverse Power Method.

```
#include <solver.hpp>
```

Collaboration diagram for InversePowerMethodSolver< MatrixType, T >:



Public Member Functions

- InversePowerMethodSolver (Input &mInputFile)

  Constructor.
- T solve () const override

  Solve the eigenvalue problem using the Inverse Power Method with Shift.

Additional Inherited Members

5.3.1 Detailed Description

template<typename MatrixType, typename T>
class InversePowerMethodSolver< MatrixType, T >

Solver for computing an eigenvalue close to a given shift using the Inverse Power Method.

The Inverse Power Method isolates and computes an eigenvalue near a user-defined shift value. This solver is particularly useful for finding eigenvalues other than the dominant one. Convergence is determined by a tolerance threshold and a maximum number of iterations.

Template Parameters

MatrixType	The type of the matrix (Eigen::MatrixXd for real or Eigen::MatrixXcd for complex).
T	The type of the eigenvalue (double for real or std::complex<double> for complex).

Note

If the shifted matrix is singular or nearly singular, the method may fail.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 InversePowerMethodSolver()

```
template<typename MatrixType , typename T >
InversePowerMethodSolver< MatrixType, T >::InversePowerMethodSolver (
            Input & mInputFile )  [inline], [explicit]
```

Constructor.

5.3.3 Member Function Documentation

5.3.3.1 solve()

```
template<typename MatrixType , typename T >
T InversePowerMethodSolver< MatrixType, T >::solve  [override], [virtual]
```

Solve the eigenvalue problem using the Inverse Power Method with Shift.

Compute any eigenvalue using the Inverse Power Method with a specified shift.

This method applies a shift to the matrix and solves a linear system iteratively to find an eigenvalue near the specified shift.

Returns

An eigenvalue of type T (double or std::complex<double>).

Exceptions

std::runtime_error	If the shifted matrix is singular or the method fails to converge.

Implements Solver< MatrixType, T >.

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

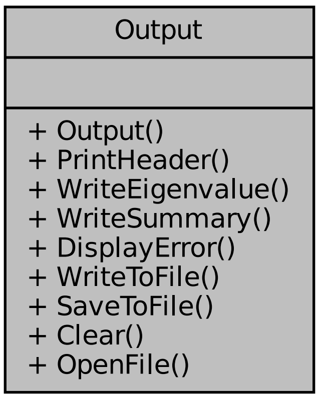
- src/solver.hpp
- src/solver_imp.hpp

5.4 Output Class Reference

Handles formatted output of eigenvalue solver results.

`#include <output.hpp>`

Collaboration diagram for Output:



Public Member Functions

- Output (std::string fileName="output.txt")

  Constructor.
- void PrintHeader (const std::string &title)

  Displays a styled header.
- template<typename T >
  void WriteEigenvalue (const T &eigenvalue, const std::string &method)

  Writes eigenvalues.
- void WriteSummary (const Input &inputFile)

  Writes a summary of the input parameters.
- void DisplayError (const std::string &message)

  Displays error messages.
- void WriteToFile (const std::string &message)

  Appends messages to the output file.
- void SaveToFile () const

  Saves the output to the specified file.
- void Clear ()

  Clears the output buffer.
- void OpenFile () const

  Opens the output file with the system's default application.

5.4.1 Detailed Description

Handles formatted output of eigenvalue solver results.

The `Output` class provides functionality to display and save results of eigenvalue solvers. It includes methods for:

- Displaying headers and results in a styled format on the console.
- Writing results and errors to a file.
- Managing formatted output for real and complex eigenvalues.

Note

ANSI escape codes are used for terminal styling, which may not be supported in all environments.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Output()

```
Output::Output (
            std::string fileName = "output.txt" )  [inline], [explicit]
```

Constructor.

5.4.3 Member Function Documentation

5.4.3.1 Clear()

```
void Output::Clear ( )  [inline]
```

Clears the output buffer.

5.4.3.2 DisplayError()

```
void Output::DisplayError (
             const std::string & message )  [inline]
```

Displays error messages.

Displays an error message in a styled format on the console and logs it to the output file.

Parameters

message	The error message to display in the terminal and log in the output file.

References BOLD, centerText(), RED, and RESET.

5.4.3.3 OpenFile()

```
void Output::OpenFile ( ) const  [inline]
```

Opens the output file with the system's default application.

Opens the output file.

5.4.3.4 PrintHeader()

```
void Output::PrintHeader (
             const std::string & title )  [inline]
```

Displays a styled header.

Displays a styled header with a title.

Parameters

title	The title to display.

References BOLD, centerText(), MAGENTA, and RESET.

5.4.3.5 SaveToFile()

```
void Output::SaveToFile ( ) const  [inline]
```

Saves the output to the specified file.

Writes eigenvalue(s) to both the terminal and the output file.

Template Parameters

T	The type of eigenvalue (scalar, complex, or vector).

/**

Saves the collected output to the specified file.

5.4.3.6 WriteEigenvalue()

```
template<typename T >
void Output::WriteEigenvalue (
             const T & eigenvalue,
             const std::string & method )
```

Writes eigenvalues.

Writes eigenvalue(s) to both the terminal and the output file.

Template Parameters

T	The type of eigenvalue (scalar, complex, or vector).

Parameters

eigenvalue	The computed eigenvalue(s).
method	The name of the method used to compute the eigenvalue(s).

References centerText(), complexToString(), CYAN, and RESET.

5.4.3.7 WriteSummary()

```
void Output::WriteSummary (
             const Input & inputFile )  [inline]
```

Writes a summary of the input parameters.

Writes a summary of input parameters to the output.

References centerText(), Input::GetComplexMatrix(), Input::GetDimension(), Input::GetIsComplex(), Input::Get←MaxIter(), Input::GetMethod(), Input::GetRealMatrix(), Input::GetShift(), Input::GetTolerance(), MAGENTA, and RESET.

5.4.3.8 WriteToFile()

```
void Output::WriteToFile (
             const std::string & message )  [inline]
```

Appends messages to the output file.

Logs a message to the output file, including eigenvalues or error details.

Parameters

message	The message to log in the output file.

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

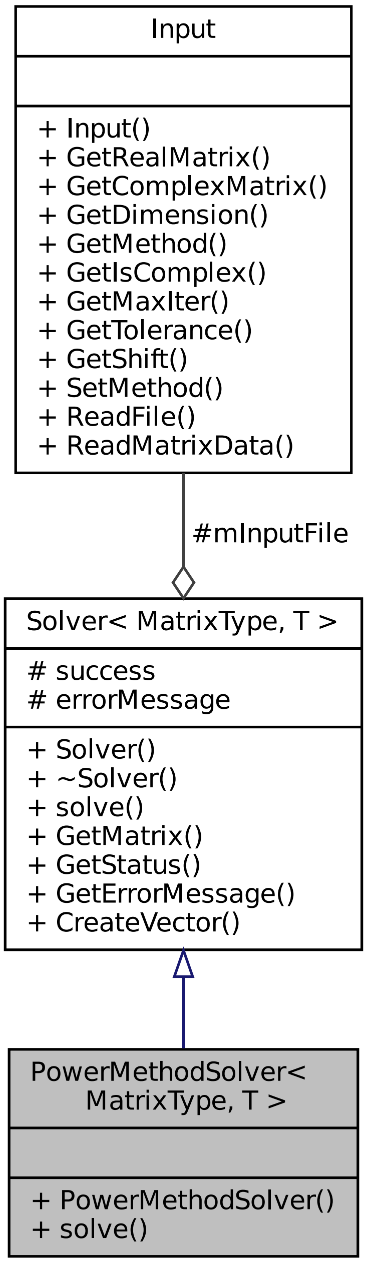
- src/output.hpp
- src/output_imp.hpp

5.5 PowerMethodSolver< MatrixType, T > Class Template Reference

Solver for computing the dominant eigenvalue using the Power Method.

```
#include <solver.hpp>
```

Collaboration diagram for PowerMethodSolver< MatrixType, T >:



Public Member Functions

- PowerMethodSolver (Input &mInputFile)

    Constructor.
- T solve () const override

    Solve the eigenvalue problem using the Power Method.

Additional Inherited Members

5.5.1 Detailed Description

template<typename MatrixType, typename T>
class PowerMethodSolver< MatrixType, T >

Solver for computing the dominant eigenvalue using the Power Method.

The Power Method iteratively computes the dominant eigenvalue (the eigenvalue with the largest absolute value) and its corresponding eigenvector for a given square matrix. This solver is suitable for real and complex matrices and relies on a tolerance threshold and a maximum number of iterations to determine convergence.

Template Parameters

MatrixType	The type of the matrix (Eigen::MatrixXd for real or Eigen::MatrixXcd for complex).
T	The type of the eigenvalue (double for real or std::complex<double> for complex).

5.5.2 Constructor & Destructor Documentation

5.5.2.1 PowerMethodSolver()

```
template<typename MatrixType , typename T >
PowerMethodSolver< MatrixType, T >::PowerMethodSolver (
            Input & mInputFile )  [inline], [explicit]
```

Constructor.

5.5.3 Member Function Documentation

5.5.3.1 solve()

```
template<typename MatrixType , typename T >
T PowerMethodSolver< MatrixType, T >::solve  [override], [virtual]
```

Solve the eigenvalue problem using the Power Method.

Compute the dominant eigenvalue using the Power Method.

This iterative method estimates the largest eigenvalue of the matrix by repeatedly multiplying a vector with the matrix. The eigenvalue is computed using the Rayleigh quotient.

Returns

The dominant eigenvalue of type T (double or std::complex<double>).

Implements Solver< MatrixType, T >.

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

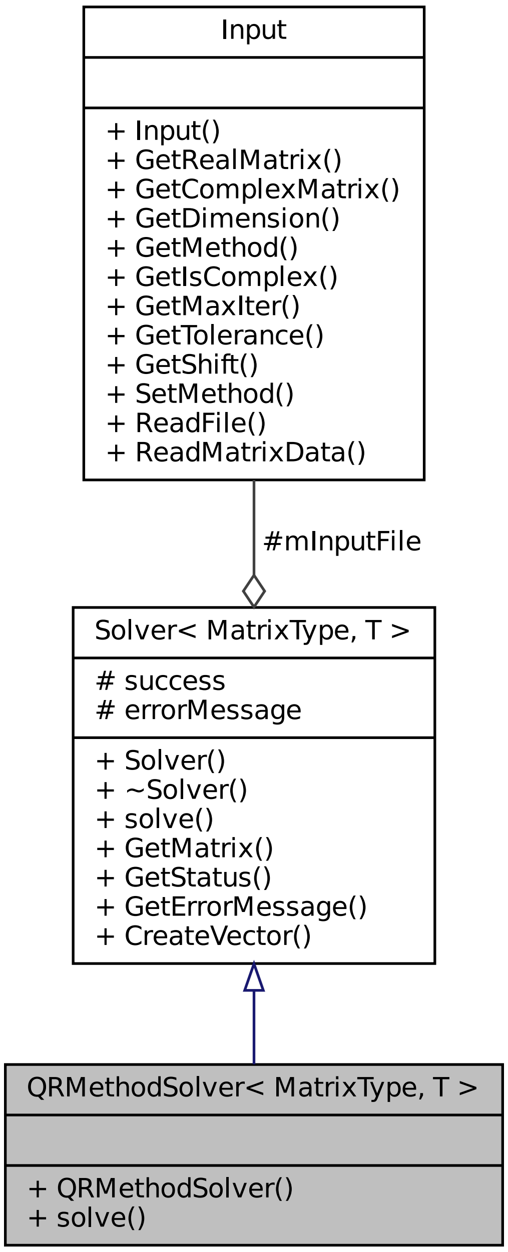
- src/solver.hpp
- src/solver_imp.hpp

5.6 QRMethodSolver< MatrixType, T > Class Template Reference

Solver for computing all eigenvalues using the QR Algorithm.

```
#include <solver.hpp>
```

Collaboration diagram for QRMethodSolver< MatrixType, T >:



Public Member Functions

- QRMethodSolver (Input &mInputFile)

  Constructor.
- T solve () const override

  Solve the eigenvalue problem using the QR algorithm.

Additional Inherited Members

5.6.1 Detailed Description

template<typename MatrixType, typename T>
class QRMethodSolver< MatrixType, T >

Solver for computing all eigenvalues using the QR Algorithm.

The QR Algorithm decomposes a given square matrix into Q (orthogonal) and R (upper triangular) matrices iteratively. It computes all eigenvalues of the matrix by transforming it into an upper triangular form, where the diagonal entries represent the eigenvalues.

Template Parameters

MatrixType	The type of the matrix (Eigen::MatrixXd for real or Eigen::MatrixXcd for complex).
T	The type of the eigenvalues (Eigen::VectorXd for real or Eigen::VectorXcd for complex).

5.6.2 Constructor & Destructor Documentation

5.6.2.1 QRMethodSolver()

```
template<typename MatrixType , typename T >
QRMethodSolver< MatrixType, T >::QRMethodSolver (
            Input & mInputFile )  [inline], [explicit]
```

Constructor.

5.6.3 Member Function Documentation

5.6.3.1 solve()

```
template<typename MatrixType , typename T >
T QRMethodSolver< MatrixType, T >::solve  [override], [virtual]
```

Solve the eigenvalue problem using the QR algorithm.

Compute all eigenvalues using the QR Algorithm.

This iterative algorithm uses QR decomposition to transform the matrix into an upper triangular form. The eigenvalues are then extracted from the diagonal of the resulting matrix.

Returns

A vector containing all eigenvalues of type T (double or std::complex<double>).

Implements Solver< MatrixType, T >.

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

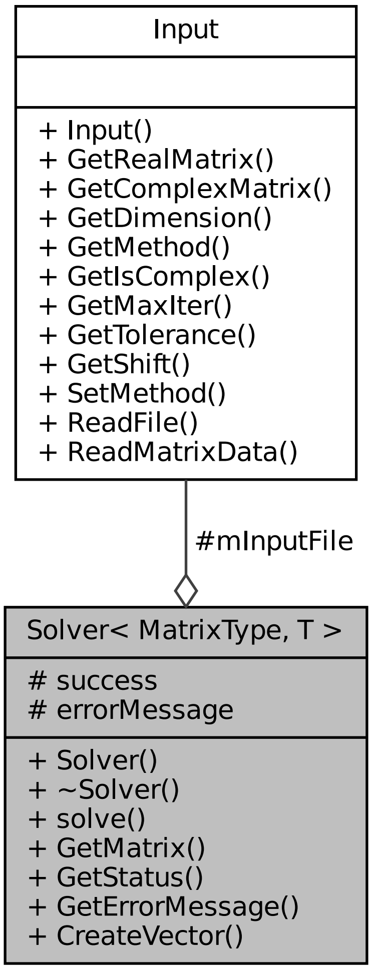
- src/solver.hpp
- src/solver_imp.hpp

5.7 Solver< MatrixType, T > Class Template Reference

Base class for eigenvalue solvers.

```
#include <solver.hpp>
```

Collaboration diagram for Solver< MatrixType, T >:



Public Member Functions

- Solver (Input &mInputFile)

    Constructor taking an Input object reference.
- virtual ∼Solver ()=default

    Virtual destructor.
- virtual T solve () const =0

    Pure virtual method for solving the eigenvalue problem.
- MatrixType GetMatrix () const

    Retrieve the matrix from the input file.
- bool GetStatus () const

    Returns the success flag.
- std::string GetErrorMessage () const

    Returns the error message.
- VectorType< MatrixType >::type CreateVector () const

    Create vector initialized to ones.

Protected Attributes

- Input & mInputFile

  Reference to the Input object.
- bool success = true

  Status of computation success.
- std::string errorMessage

  Stores error messages.

5.7.1 Detailed Description

template<typename MatrixType, typename T>
class Solver< MatrixType, T >

Base class for eigenvalue solvers.

Provides the interface and common functionality for eigenvalue solvers.

Template Parameters

MatrixType	The type of the matrix (real or complex).
T	The type of the eigenvalue (real or complex).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Solver()

```
template<typename MatrixType , typename T >
Solver< MatrixType, T >::Solver (
             Input & mInputFile )  [inline], [explicit]
```

Constructor taking an Input object reference.

5.7.2.2 ∼Solver()

```
template<typename MatrixType , typename T >
virtual Solver< MatrixType, T >::∼Solver ( )  [virtual], [default]
```

Virtual destructor.

5.7.3 Member Function Documentation

5.7.3.1 CreateVector()

```
template<typename MatrixType , typename T >
VectorType< MatrixType >::type Solver< MatrixType, T >::CreateVector
```

Create vector initialized to ones.

Create a vector of the appropriate type, initialized to ones.

The vector type is determined using the `VectorType` trait based on the matrix type.

Returns

A vector of type `VectorType<MatrixType>::type`, initialized with ones.

5.7.3.2 GetErrorMessage()

```
template<typename MatrixType , typename T >
std::string Solver< MatrixType, T >::GetErrorMessage ( ) const  [inline]
```

Returns the error message.

References Solver< MatrixType, T >::errorMessage.

5.7.3.3 GetMatrix()

```
template<typename MatrixType , typename T >
MatrixType Solver< MatrixType, T >::GetMatrix
```

Retrieve the matrix from the input file.

5.7.3.4 GetStatus()

```
template<typename MatrixType , typename T >
bool Solver< MatrixType, T >::GetStatus ( ) const  [inline]
```

Returns the success flag.

References Solver< MatrixType, T >::success.

5.7.3.5 solve()

```
template<typename MatrixType , typename T >
virtual T Solver< MatrixType, T >::solve ( ) const  [pure virtual]
```

Pure virtual method for solving the eigenvalue problem.

Implemented in QRMethodSolver< MatrixType, T >, InversePowerMethodSolver< MatrixType, T >, and PowerMethodSolver< MatrixType, T >.

5.7.4 Member Data Documentation

5.7.4.1 errorMessage

```
template<typename MatrixType , typename T >
std::string Solver< MatrixType, T >::errorMessage  [protected]
```

Stores error messages.

5.7.4.2 mInputFile

```
template<typename MatrixType , typename T >
Input& Solver< MatrixType, T >::mInputFile  [protected]
```

Reference to the Input object.

5.7.4.3 success

```
template<typename MatrixType , typename T >
bool Solver< MatrixType, T >::success = true  [protected]
```

Status of computation success.

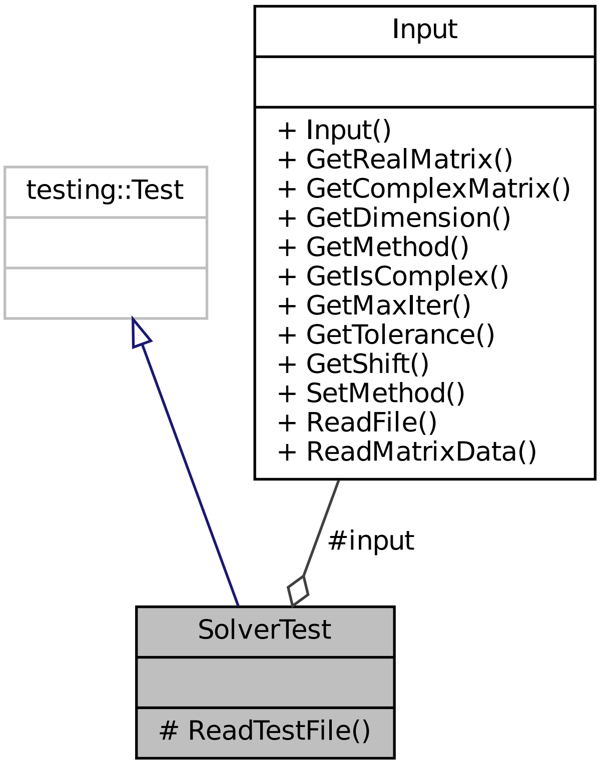
The documentation for this class was generated from the following files:

- src/solver.hpp
- src/solver_imp.hpp

5.8 SolverTest Class Reference

Google Test fixture for testing various solver classes.

Collaboration diagram for SolverTest:



Protected Member Functions

- void ReadTestFile (const std::string &fileName, const std::string &content)

  Helper function to read and parse a temporary test file.

Protected Attributes

- Input input

  Instance of `Input` for testing solvers.

5.8.1 Detailed Description

Google Test fixture for testing various solver classes.

5.8.2 Member Function Documentation

5.8.2.1 ReadTestFile()

```
void SolverTest::ReadTestFile (
             const std::string & fileName,
             const std::string & content )  [inline], [protected]
```

Helper function to read and parse a temporary test file.

Parameters

fileName	Name of the file to read.
content	Content to write to the file.

References CleanUpTempFile(), input, Input::ReadFile(), and WriteToTempFile().

5.8.3 Member Data Documentation

5.8.3.1 input

```
Input SolverTest::input  [protected]
```

Instance of `Input` for testing solvers.

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

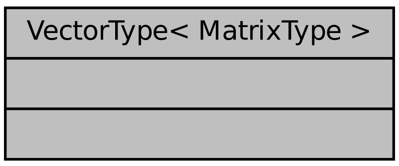
- test/test.cpp

5.9 VectorType< MatrixType > Struct Template Reference

Type trait to deduce the vector type based on the matrix type.

`#include <solver.hpp>`

Collaboration diagram for VectorType< MatrixType >:



5.9.1 Detailed Description

template<typename MatrixType>
struct VectorType< MatrixType >

Type trait to deduce the vector type based on the matrix type.

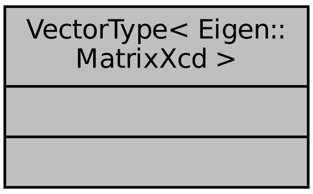
The documentation for this struct was generated from the following file:

- src/solver.hpp

5.10 VectorType< Eigen::MatrixXcd > Struct Reference

```
#include <solver.hpp>
```

Collaboration diagram for VectorType< Eigen::MatrixXcd >:



Public Types

- using type = Eigen::VectorXcd

5.10.1 Member Typedef Documentation

5.10.1.1 type

```
using VectorType< Eigen::MatrixXcd >::type = Eigen::VectorXcd
```

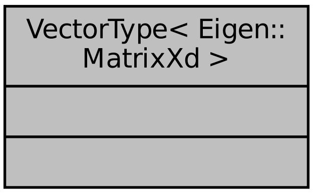
The documentation for this struct was generated from the following file:

- src/solver.hpp

5.11 VectorType< Eigen::MatrixXd > Struct Reference

```
#include <solver.hpp>
```

Collaboration diagram for VectorType< Eigen::MatrixXd >:



Public Types

- using type = Eigen::VectorXd

5.11.1 Member Typedef Documentation

5.11.1.1 type

```
using VectorType< Eigen::MatrixXd >::type = Eigen::VectorXd
```

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

- src/solver.hpp

Chapter 6

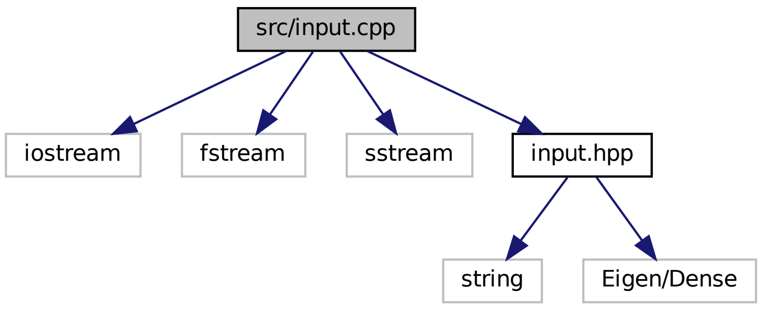
File Documentation

6.1 src/input.cpp File Reference

Implementation of the Input class for managing numerical parameters and matrices.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "input.hpp"
```

Include dependency graph for input.cpp:



6.1.1 Detailed Description

Implementation of the Input class for managing numerical parameters and matrices.

This source file provides the implementation of the `Input` class declared in `input.hpp`. It includes methods for reading and validating numerical parameters and matrix data from a file, as well as parsing real and complex matrices.

Key Implementation Details:

- Uses the Eigen library to store and manipulate matrices.
- Performs input validation for numerical parameters and matrix data.
- Supports flexible file formats with comments and whitespace handling.

Example Input File Format:

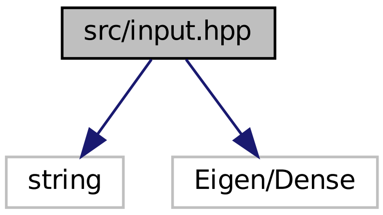
```
MAX_ITER 500
TOLERANCE 1e-10
SHIFT 0.5
COMPLEX 0
METHOD qr
MATRIX_DATA
3 3
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0
```

6.2 src/input.hpp File Reference

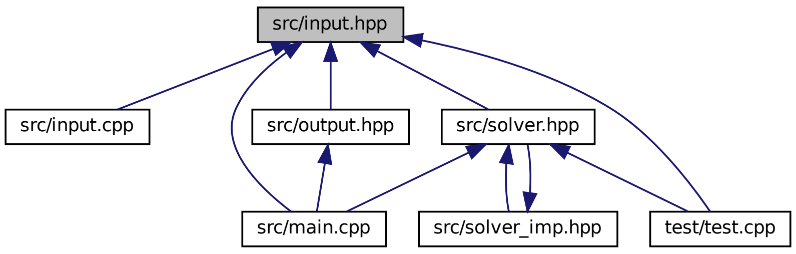
Declaration of the Input class for managing numerical parameters and matrices from a file.

```
#include <string>
#include <Eigen/Dense>
```

Include dependency graph for input.hpp:



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



Classes

- class Input

  Class to handle numerical input parameters and matrices from a file.

6.2.1 Detailed Description

Declaration of the Input class for managing numerical parameters and matrices from a file.

This header file defines the `Input` class, which provides methods to read and store configuration parameters and matrix data for numerical computations. It supports both real and complex matrices.

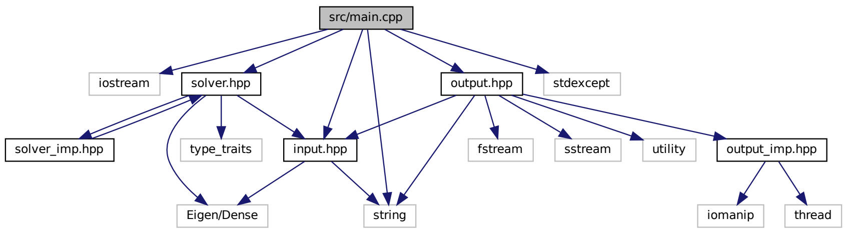
Key Features:

- Reads numerical parameters like tolerance, maximum iterations, shift, and method.
- Parses square matrices from an input file in a defined format.
- Supports validation of parameters and matrix data.

6.3 src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include <stdexcept>
#include "input.hpp"
#include "solver.hpp"
#include "output.hpp"
```

Include dependency graph for main.cpp:



Functions

- template<typename SolverType , typename T >

  void SolveAndWrite (Input &inputFile, Output &outputManager, const std::string &methodName)

  Template function to execute a solver and handle results or errors.
- int main (int argc, char ∗argv[ ])

6.3.1 Function Documentation

6.3.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

References Output::DisplayError(), Input::GetIsComplex(), Input::GetMethod(), Output::OpenFile(), Output::Print←Header(), Input::ReadFile(), Output::SaveToFile(), Output::WriteSummary(), and Output::WriteToFile().

6.3.1.2 SolveAndWrite()

```
template<typename SolverType , typename T >
void SolveAndWrite (
            Input & inputFile,
            Output & outputManager,
            const std::string & methodName )
```

Template function to execute a solver and handle results or errors.

Template Parameters

SolverType	The solver class type.
T	The type of the eigenvalue result (e.g., double, std::complex<double>, or Eigen vector).

Parameters

inputFile	The input object with the problem definition.
outputManager	The output manager for logging and saving results.
methodName	The name of the method (e.g., "Power Method").

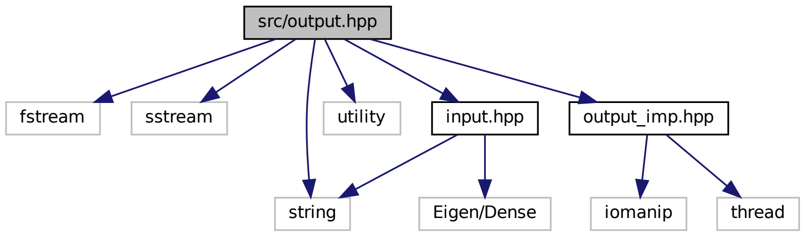
References Output::DisplayError(), Output::WriteEigenvalue(), and Output::WriteToFile().

6.4 src/output.hpp File Reference

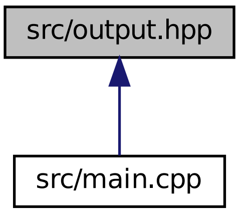
Declaration of the Output class for handling formatted output of eigenvalue solver results.

```
#include <fstream>
#include <sstream>
#include <string>
#include <utility>
#include "input.hpp"
#include "output_imp.hpp"
```

Include dependency graph for output.hpp:



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



Classes

- class Output

  Handles formatted output of eigenvalue solver results.

Macros

- #define RESET "\033[0m"
- #define GREEN "\033[32m"
- #define CYAN "\033[36m"
- #define MAGENTA "\033[35m"
- #define BOLD "\033[1m"
- #define RED "\033[31m"

6.4.1 Detailed Description

Declaration of the Output class for handling formatted output of eigenvalue solver results.

This header file defines the `Output` class, which provides methods for:

- Printing styled headers and results to the console.
- Saving results to an external file.
- Managing output formatting for both real and complex eigenvalues.
- Displaying input summaries and error messages.

The class leverages ANSI escape codes for styled terminal output and supports detailed formatting for matrices and vectors.

6.4.2 Macro Definition Documentation

6.4.2.1 BOLD

```
#define BOLD "\033[1m"
```

6.4.2.2 CYAN

```
#define CYAN "\033[36m"
```

6.4.2.3 GREEN

```
#define GREEN "\033[32m"
```

6.4.2.4 MAGENTA

```
#define MAGENTA "\033[35m"
```

6.4.2.5 RED

```
#define RED "\033[31m"
```

6.4.2.6 RESET

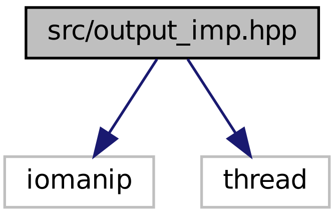
```
#define RESET "\033[0m"
```

6.5 src/output_imp.hpp File Reference

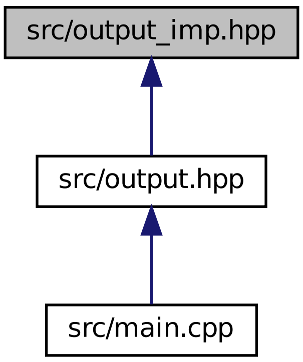
Implementation of the Output class methods for managing formatted output.

```
#include <iomanip>
#include <thread>
```

Include dependency graph for output_imp.hpp:



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



Functions

- std::string centerText (const std::string &text, int totalWidth)

  Utility function to center-align text.

- template< typename T >
  std::string complexToString (const std::complex< T > &c)

  Utility function to convert a complex number to a formatted string.

6.5.1 Detailed Description

Implementation of the Output class methods for managing formatted output.

This file provides the implementation of the `Output` class declared in `output.hpp`. It includes methods for styled console output, saving results to a file, and formatting real and complex eigenvalues.

6.5.2 Function Documentation

6.5.2.1 centerText()

```
std::string centerText (
             const std::string & text,
             int totalWidth )  [inline]
```

Utility function to center-align text.

Parameters

text	The text to center.
totalWidth	Total width of the field.

Returns

A string with the text centered.

6.5.2.2 complexToString()

```
template<typename T >
std::string complexToString (
             const std::complex< T > & c )
```

Utility function to convert a complex number to a formatted string.

Template Parameters

T	The numeric type of the complex number.

Parameters

c	The complex number to convert.

Returns

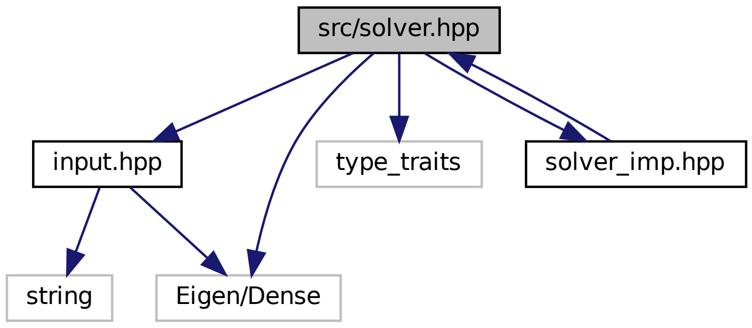
A string representation of the complex number.

6.6 src/solver.hpp File Reference

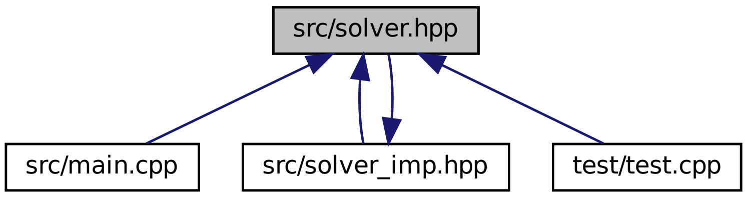
Declaration of eigenvalue solvers using Power Method, Inverse Power Method, and QR Algorithm.

```
#include "input.hpp"
#include <Eigen/Dense>
#include <type_traits>
#include "solver_imp.hpp"
```

Include dependency graph for solver.hpp:



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



Classes

- struct VectorType< MatrixType >

  Type trait to deduce the vector type based on the matrix type.
- struct VectorType< Eigen::MatrixXd >
- struct VectorType< Eigen::MatrixXcd >

- class Solver< MatrixType, T >

  Base class for eigenvalue solvers.
- class PowerMethodSolver< MatrixType, T >

  Solver for computing the dominant eigenvalue using the Power Method.
- class InversePowerMethodSolver< MatrixType, T >

  Solver for computing an eigenvalue close to a given shift using the Inverse Power Method.
- class QRMethodSolver< MatrixType, T >

  Solver for computing all eigenvalues using the QR Algorithm.

6.6.1 Detailed Description

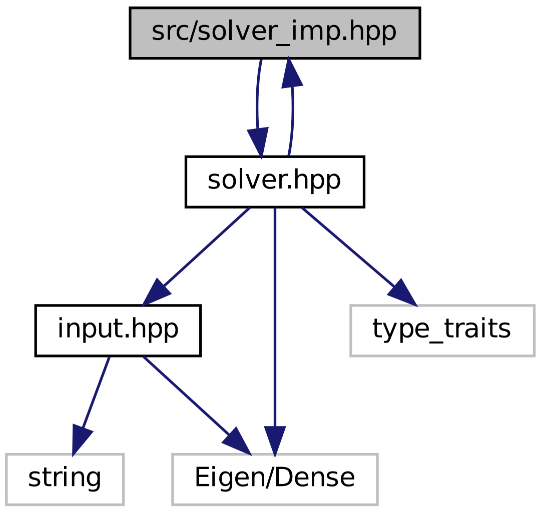
Declaration of eigenvalue solvers using Power Method, Inverse Power Method, and QR Algorithm.

Defines the abstract base class `Solver` and its derived classes for computing eigenvalues using various numerical techniques. Includes support for real and complex matrices.

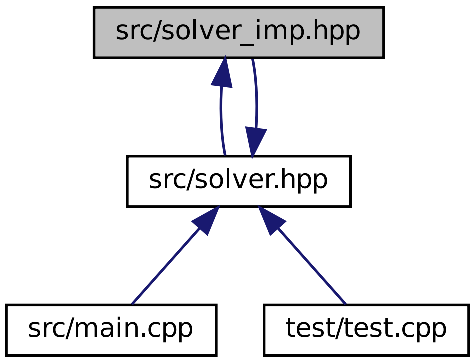
6.7 src/solver_imp.hpp File Reference

Implementation of the Solver classes and their respective eigenvalue computation methods.

```
#include "solver.hpp"
```
Include dependency graph for solver_imp.hpp:



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



6.7.1 Detailed Description

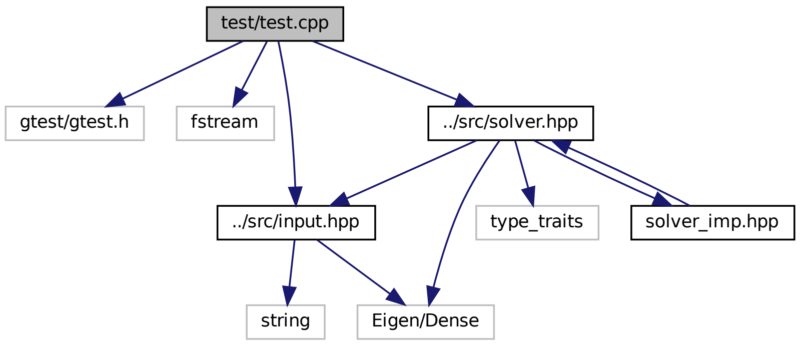
Implementation of the Solver classes and their respective eigenvalue computation methods.

This file provides the implementation of the methods declared in `solver.hpp`, including the Power Method, Inverse Power Method with Shift, and the QR Algorithm.

6.8 test/test.cpp File Reference

```
#include <gtest/gtest.h>
#include <fstream>
#include "../src/input.hpp"
#include "../src/solver.hpp"
```

Include dependency graph for test.cpp:



Classes

- class InputTest

  Google Test fixture for testing the `Input` class.
- class SolverTest

  Google Test fixture for testing various solver classes.

Functions

- void WriteToTempFile (const std::string &fileName, const std::string &content)

  Creates and writes content to a temporary file.
- void CleanUpTempFile (const std::string &fileName)

  Deletes a temporary file.
- TEST_F (InputTest, ParsesValidRealMatrix)

  Validates parsing a real matrix from input.
- TEST_F (InputTest, ParsesValidComplexMatrix)

  Validates parsing a complex matrix from input.
- TEST_F (InputTest, ThrowsForInvalidMatrixDimensions)

  Validates exception is thrown for invalid matrix dimensions.
- TEST_F (InputTest, ThrowsForInvalidMethod)

  Validates exception is thrown for invalid METHOD parameter.
- TEST_F (InputTest, ParsesSingularMatrix)

  Validates parsing a singular matrix from input.
- TEST_F (InputTest, ThrowsForMissingMatrixData)

  Validates exception is thrown when MATRIX_DATA is missing.
- TEST_F (SolverTest, PowerMethodSolverRealMatrix)

  Validates the Power Method solver with a real matrix.
- TEST_F (SolverTest, InversePowerMethodSolverRealMatrix)

  Validates the Inverse Power Method solver with a real matrix.
- TEST_F (SolverTest, QRMethodSolverRealMatrix)

  Validates the QR Method solver with a real matrix.
- TEST_F (SolverTest, PowerMethodSolverComplexMatrix)

  Validates the Power Method solver with a complex matrix.
- TEST_F (SolverTest, QRMethodSolverComplexMatrix)

  Validates the QR Method solver with a complex matrix.
- void RunSolverAndThrow (Input &input)

  Executes the Inverse Power Method Solver and propagates exceptions.
- TEST_F (SolverTest, InversePowerMethodSolverSingularRealMatrixThrows)

  Validates exception handling for singular real matrices in the Inverse Power Method solver.
- TEST_F (SolverTest, InversePowerMethodSolverSingularRealMatrixThrowsWithShift)

  Validates exception handling for singular real matrices in the Inverse Power Method solver with a non-zero shift.
- TEST_F (SolverTest, QRConvergenceTest)

  Validates the QR Method solver's convergence for a simple real matrix.

6.8.1 Function Documentation

6.8.1.1 CleanUpTempFile()

```
void CleanUpTempFile (
            const std::string & fileName )
```

Deletes a temporary file.

Parameters

fileName	Name of the file to delete.

6.8.1.2 RunSolverAndThrow()

```
void RunSolverAndThrow (
            Input & input )
```

Executes the Inverse Power Method Solver and propagates exceptions.

This function is designed to test scenarios where the solver is expected to fail (or not)due to invalid input (e.g., singular matrices). It is used to test the difference between singular matrices and the case where the file doesn't converge .

Parameters

input	An instance of the `Input` class containing solver configuration.

Exceptions

std::runtime_error	If the solver encounters an issue during execution.

References InversePowerMethodSolver< MatrixType, T >::solve().

6.8.1.3 TEST_F() `[1/14]`

```
TEST_F (
            InputTest ,
            ParsesSingularMatrix  )
```

Validates parsing a singular matrix from input.

Test InputTest::ParsesSingularMatrix

6.8.1.4 TEST_F() `[2/14]`

```
TEST_F (
            InputTest ,
            ParsesValidComplexMatrix  )
```

Validates parsing a complex matrix from input.

Test InputTest::ParsesValidComplexMatrix

6.8.1.5 TEST_F() [3/14]

```
TEST_F (
            InputTest ,
            ParsesValidRealMatrix  )
```

Validates parsing a real matrix from input.

Test InputTest::ParsesValidRealMatrix

6.8.1.6 TEST_F() [4/14]

```
TEST_F (
            InputTest ,
            ThrowsForInvalidMatrixDimensions  )
```

Validates exception is thrown for invalid matrix dimensions.

Test InputTest::ThrowsForInvalidMatrixDimensions

6.8.1.7 TEST_F() [5/14]

```
TEST_F (
            InputTest ,
            ThrowsForInvalidMethod  )
```

Validates exception is thrown for invalid METHOD parameter.

Test InputTest::ThrowsForInvalidMethod

6.8.1.8 TEST_F() [6/14]

```
TEST_F (
            InputTest ,
            ThrowsForMissingMatrixData  )
```

Validates exception is thrown when MATRIX_DATA is missing.

Test InputTest::ThrowsForMissingMatrixData

6.8.1.9 TEST_F() [7/14]

```
TEST_F (
          SolverTest ,
          InversePowerMethodSolverRealMatrix  )
```

Validates the Inverse Power Method solver with a real matrix.

Test SolverTest::InversePowerMethodSolverRealMatrix

Tests that the solver computes the eigenvalue closest to a given shift. < Eigenvalue closest to shift

References InversePowerMethodSolver< MatrixType, T >::solve().

6.8.1.10 TEST_F() [8/14]

```
TEST_F (
          SolverTest ,
          InversePowerMethodSolverSingularRealMatrixThrows  )
```

Validates exception handling for singular real matrices in the Inverse Power Method solver.

Test SolverTest::InversePowerMethodSolverSingularRealMatrixThrows

Tests that the solver throws a runtime error when solving a singular matrix with a shift of 0.

References RunSolverAndThrow().

6.8.1.11 TEST_F() [9/14]

```
TEST_F (
          SolverTest ,
          InversePowerMethodSolverSingularRealMatrixThrowsWithShift  )
```

Validates exception handling for singular real matrices in the Inverse Power Method solver with a non-zero shift.

Test SolverTest::InversePowerMethodSolverSingularRealMatrixThrowsWithShift

Ensures the solver throws a runtime error even when a shift (e.g., 2.0) is applied.

References RunSolverAndThrow().

6.8.1.12 TEST_F() [10/14]

```
TEST_F (
            SolverTest ,
            PowerMethodSolverComplexMatrix  )
```

Validates the Power Method solver with a complex matrix.

Test SolverTest::PowerMethodSolverComplexMatrix

Ensures the solver correctly computes the largest eigenvalue with both real and imaginary parts. < Real part of the largest eigenvalue.

< Imaginary part of the largest eigenvalue.

References PowerMethodSolver< MatrixType, T >::solve().

6.8.1.13 TEST_F() [11/14]

```
TEST_F (
            SolverTest ,
            PowerMethodSolverRealMatrix  )
```

Validates the Power Method solver with a real matrix.

Test SolverTest::PowerMethodSolverRealMatrix

Tests that the solver correctly computes the largest eigenvalue. < Largest eigenvalue

References PowerMethodSolver< MatrixType, T >::solve().

6.8.1.14 TEST_F() [12/14]

```
TEST_F (
            SolverTest ,
            QRConvergenceTest  )
```

Validates the QR Method solver's convergence for a simple real matrix.

Test SolverTest::QRConvergenceTest

Ensures the solver runs without throwing any errors when handling a non-singular matrix.

References RunSolverAndThrow().

6.8.1.15 TEST_F() [13/14]

```
TEST_F (
            SolverTest ,
            QRMethodSolverComplexMatrix  )
```

Validates the QR Method solver with a complex matrix.

Test SolverTest::QRMethodSolverComplexMatrix

Ensures the solver computes all eigenvalues accurately. < Ensure correct result size.

References QRMethodSolver< MatrixType, T >::solve().

6.8.1.16 TEST_F() [14/14]

```
TEST_F (
            SolverTest ,
            QRMethodSolverRealMatrix  )
```

Validates the QR Method solver with a real matrix.

Test SolverTest::QRMethodSolverRealMatrix

Ensures the solver computes all eigenvalues accurately. < Ensure correct result size.

< Largest eigenvalue.

< Middle eigenvalue.

< Smallest eigenvalue.

References QRMethodSolver< MatrixType, T >::solve().

6.8.1.17 WriteToTempFile()

```
void WriteToTempFile (
            const std::string & fileName,
            const std::string & content )
```

Creates and writes content to a temporary file.

Parameters

fileName	Name of the file to create.
content	Content to write to the file.

Exceptions

std::runtime_error	If the file cannot be created.

Index