My Project

Generated by Doxygen 1.8.6

Wed Aug 8 2018 12:59:48

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 ConstantGridSolver Class Reference

`#include <ConstantGridSolver.h>`

Public Member Functions

- arma::cx_mat calculateT (int j, double E) const

  Calculates $\mathbf{T}(x_j, E)$ matrix.
- arma::cx_mat calculateU (int j, double E)

  Calculates $\mathbf{U}(x_j, E)$ matrix.
- arma::cx_mat calculateEP (int j, double E)

  Calculates $\mathbf{E}^+(x_j, E)$ matrix.
- arma::cx_mat calculateEM (int j, double E)

  Calculates $\mathbf{E}^-(x_j, E)$ matrix.
- void modifyCCnj (arma::cx_mat &n1, arma::cx_mat &n0, arma::cx_mat &j1, arma::cx_mat &j0, double E)

  Modifies closed channels elements.
- arma::cx_mat fwdIteration (const arma::cx_mat &B, double E)

  Iterates Numerov algorithm forward up to $N-1$ and returns $\mathbf{R}_{N-1}$ matrix for a given energy.
- arma::cx_mat calculateS (const arma::cx_mat R_N, double E)

  Calculates $\mathbf{S}$ matrix for given $\mathbf{R}_{N-1}$.
- void saveS (const arma::cx_mat &S, const int E, const std::string directory)

  Saves $\mathbf{S}$ matrix (Im and Re part separately).
- void setParameters (const Parameters &parameters)
- ConstantGridSolver ()=default
- ∼ConstantGridSolver ()=default
- ConstantGridSolver (const Parameters &params)

  Constructor.
- void solveForEnergies (std::string directory)

  Performs Numerov calculations for a given set of parameters for all energies.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 ConstantGridSolver::ConstantGridSolver ( ) `[default]`

3.1.1.2 ConstantGridSolver::∼ConstantGridSolver ( ) `[default]`

3.1.1.3 ConstantGridSolver::ConstantGridSolver (const Parameters & *params*) [explicit]

Constructor.

3.1.2 Member Function Documentation

3.1.2.1 arma::cx_mat ConstantGridSolver::calculateEM (int *j*, double *E*)

Calculates $\mathbf{E}^{-}(x_j, E)$ matrix.

This method calculates $\mathbf{E}^{-}$ matrix for for a given point x_j on the grid and given energy. The matrix is diagonal and its elements are calculated the following way:

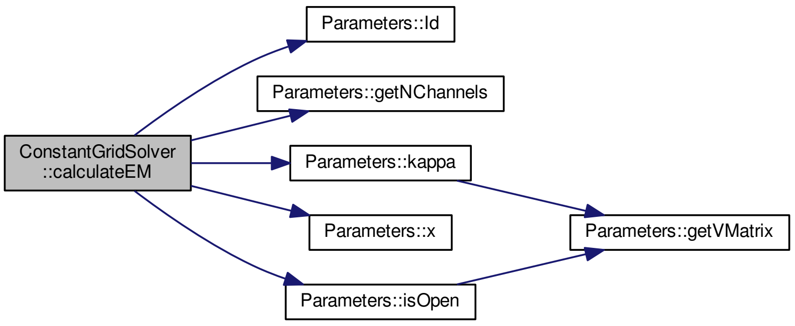
- $\mathbf{E}^{-}_{n,n}(x_j, E) = \exp(-ikx_j)$ if channel n is open
- $\mathbf{E}^{-}_{n,n}(x_j, E) = \cosh(kx_j)$ if channel n is closed
- $\mathbf{E}^{-}_{n,m}(x_j, E) = 0$ for $n \neq m$

Parameters

j	
E	

Returns

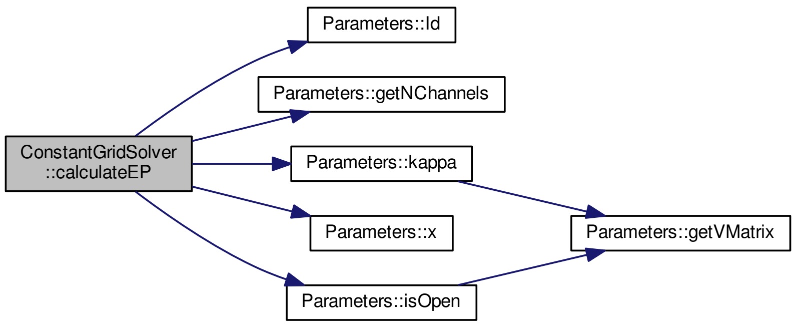
Here is the call graph for this function:



3.1.2.2 arma::cx_mat ConstantGridSolver::calculateEP (int *j*, double *E*)

Calculates $\mathbf{E}^{+}(x_j, E)$ matrix.

Here is the call graph for this function:



3.1.2.3 arma::cx_mat ConstantGridSolver::calculateS (const arma::cx_mat *R_N*, double *E*)

Calculates $\mathbf{S}$ matrix for given $\mathbf{R}_{N-1}$.

This method calculates the scattering matrix $\mathbf{S}(E)$. Its value is given by

$$\mathbf{S} = (\mathbf{R}_{N-1}\mathbf{e}^{+}_{N-1} - \mathbf{e}^{+}_{N}) - 1(\mathbf{R}_{N-1}\mathbf{e}^{-}_{N-1} - \mathbf{e}^{-}_{N}) \tag{3.1}$$

where $\mathbf{e}^{\pm}_{i} = (\mathbf{I} - \mathbf{T}_i)\mathbf{E}^{\pm}_{i}$.

Parameters

R_N	- $\mathbf{R}_{N-1}$
E	- energy

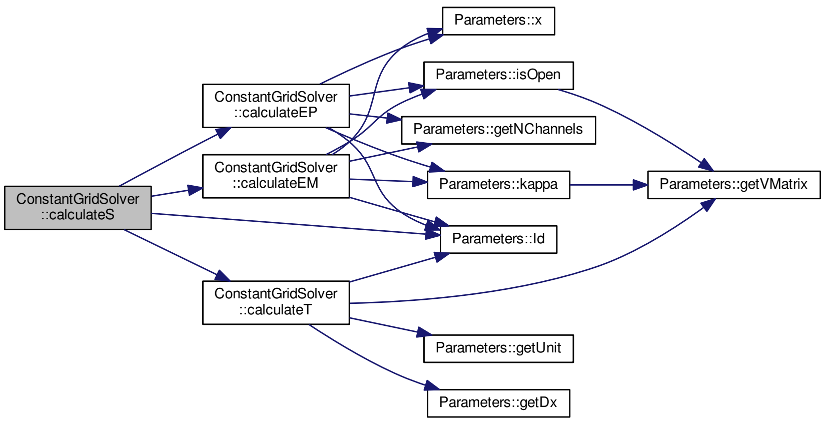
Returns

$\mathbf{S}$

Exceptions

std::runtime_error	if there is a problem with calculating

Here is the call graph for this function:



3.1.2.4 arma::cx_mat ConstantGridSolver::calculateT (int *j*, double *E*) const

Calculates $\mathbf{T}(x_j,E)$ matrix.

This method calculates $\mathbf{T}$ matrix for a given point x_j on the grid and given energy according to the formula:

$$\mathbf{T}_j = -\frac{dx}{12}\mathbf{Q}_j. \tag{3.2}$$

Parameters

j	- index of the value on the grid
E	- energy

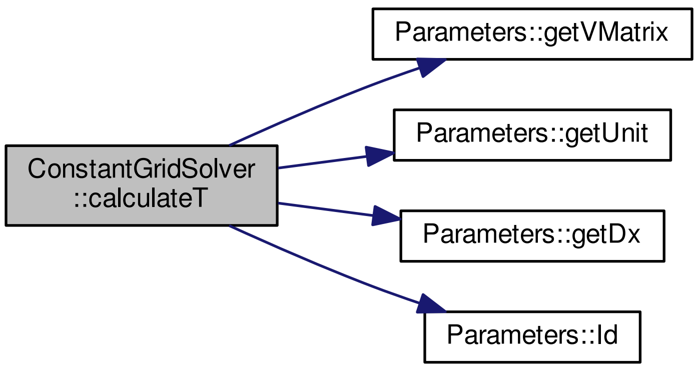
Returns

$\mathbf{T}(x_j,E)$

Exceptions

std::invalid_argument	if the index *j* is wrong

Here is the call graph for this function:



3.1.2.5 arma::cx_mat ConstantGridSolver::calculateU (int *j*, double *E*)

Calculates $\mathbf{U}(x_j, E)$ matrix.

This method calculates $\mathbf{U}$ matrix for given index j using the set of parameters provided to the ConstantGridSolver object according to the following formula:

$$\mathbf{U}_j = 12(\mathbf{I} - \mathbf{T}_j) {-1} - 10\mathbf{I}. \tag{3.3}$$

Parameters

in	*j*	- index on the grid of x value
in	*E*	- energy value

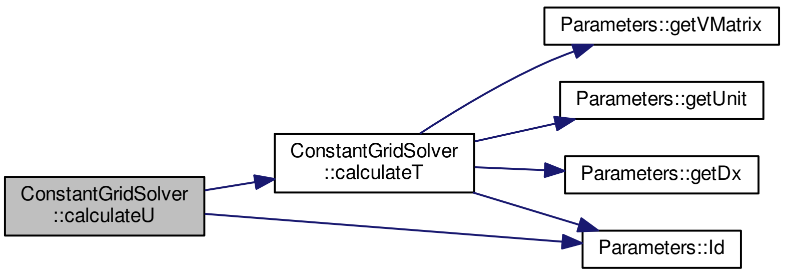
Returns

$\mathbf{U}(x_j, E)$

Exceptions

std::invalid_argument	if x_j does not exist

Here is the call graph for this function:



3.1.2.6 arma::cx_mat ConstantGridSolver::fwdIteration (const arma::cx_mat & *B*, double *E*)

Iterates Numerov algorithm forward up to $N-1$ and returns $\mathbf{R}_{N-1}$ matrix for a given energy.

This method performs the Numerov iteration for a given energy for a case of some particular symmetry.

The initial value $\mathbf{R}_0^{-1}$:

- $\mathbf{R}_0^{-1} = \mathbf{0}$ if no symmetries
- $\mathbf{R}_0^{-1} = \mathbf{U}_0^{-1}(\mathbf{I}+\mathbf{B})$ if the symmetry is described by $\mathbf{B}$

Every value depends on the previous one: $\mathbf{R}_j = \mathbf{U}_j - \mathbf{R}_{j-1}^{-1}$.

Parameters

`in`	*B*	- $\mathbf{B}$ matrix to calculate the initial value
`in`	*E*	- energy

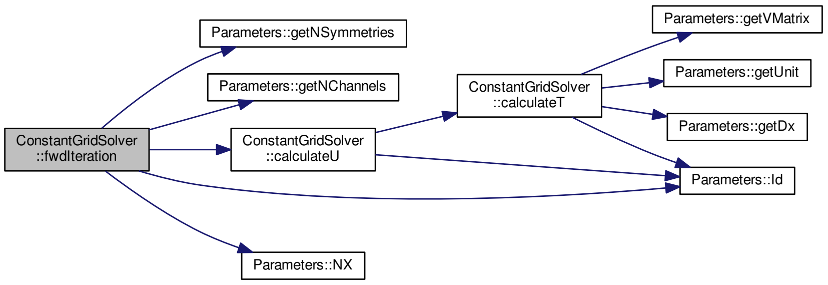
Returns

$\mathbf{R}_{N-1}$

Exceptions

std::invalid_argument	if $\mathbf{U}_i$ cannot be calculated for given iteration i
std::runtime_error	

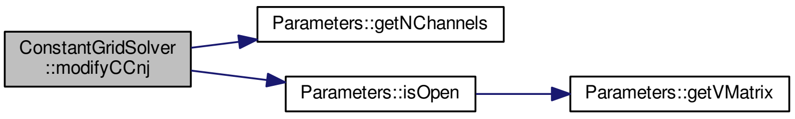
Here is the call graph for this function:


ConstantGridSolver ::fwdIteration
Parameters::getNSymmetries
Parameters::getNChannels
ConstantGridSolver ::calculateU
ConstantGridSolver ::calculateT
Parameters::getVMatrix
Parameters::getUnit
Parameters::getDx
Parameters::Id
Parameters::NX

3.1.2.7 void ConstantGridSolver::modifyCCnj (arma::cx_mat & *n1,* arma::cx_mat & *n0,* arma::cx_mat & *j1,* arma::cx_mat & *j0,* double *E*)

Modifies closed channels elements.

Here is the call graph for this function:



3.1.2.8 void ConstantGridSolver::saveS (const arma::cx_mat & *S,* const int *E,* const std::string *directory*)

Saves $\mathbf{S}$ matrix (Im and Re part separately).

This method saves the scattering matrix in a given directory. The real and imaginary part of $\mathbf{S}$ are saved in separate files.

Paths:

$Re(\mathbf{S})$: directory/re_SE.dat (E is the value of the energy) $Im(\mathbf{S})$: directory/im_SE.dat (E is the value of the energy)

Parameters

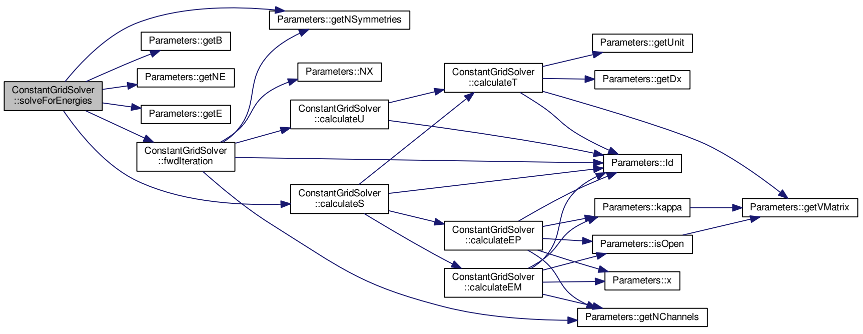
S	- scattering matrix to be saved
E	- energy
directory	- where to save the files

3.1.2.9 void ConstantGridSolver::setParameters (const Parameters & *parameters*) `[inline]`

3.1.2.10 void ConstantGridSolver::solveForEnergies (std::string *directory*)

Performs Numerov calculations for a given set of parameters for all energies.

Here is the call graph for this function:


ConstantGridSolver ::solveForEnergies
Parameters::getNSymmetries
Parameters::getB
Parameters::getNE
Parameters::getE
ConstantGridSolver ::fwdIteration
Parameters::NX
ConstantGridSolver ::calculateU
ConstantGridSolver ::calculateS
ConstantGridSolver ::calculateT
ConstantGridSolver ::calculateEP
ConstantGridSolver ::calculateEM
Parameters::getUnit
Parameters::getDx
Parameters::Id
Parameters::kappa
Parameters::isOpen
Parameters::x
Parameters::getNChannels
Parameters::getVMatrix

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

- ConstantGridSolver.h
- ConstantGridSolver.cpp

3.2 Parameters Class Reference

```
#include <Parameters.h>
```

Public Member Functions

- void loadParams (std::string="Params.txt")

    Reading the values of parameters from the file generated in Mathematica.
- void setXValues ()

    Setting xValues.
- FRIEND_TEST (ParametersInputTest, setXValues_failsIfXMaxLessOrEqualXMin)
- FRIEND_TEST (ParametersInputTest, setXValues_failsIfInvalidDX)
- FRIEND_TEST (ParametersInputTest, setXValues_failsIfInvalidCombinationOfXMinXMaxDx)
- FRIEND_TEST (ParametersInputTest, setXValues_worksGoodForCorrectValues)
- void loadE (std::string filename="E.dat")

    Reading the values of energies from the file generated in Mathematica.
- void loadV (std::string filename="V.dat")

    Reading the values of V from the file generated in Mathematica.
- FRIEND_TEST (Parameters_loadV_Test, failsForIncorrectNumberOfRows)
- FRIEND_TEST (Parameters_loadV_Test, worksGoodForGoodFileOneChannel)
- FRIEND_TEST (Parameters_loadV_Test, worksGoodForGoodFileTwoChannels)

- void loadB (std::string filename="B")

  Reading the values of B from the file generated in Mathematica.
- FRIEND_TEST (ParametersInputTest, loadB_failsIfAnyFileDoesNotExistAndPositiveNSymmetries)
- FRIEND_TEST (ParametersInputTest, loadB_worksGoodForGoodFilesOneChannel)
- FRIEND_TEST (ParametersInputTest, loadB_failsIfAnyFileIsIncorrect)
- bool checkNumberOfRowsInFile (std::string filename, const int required_number_of_columns)
- Parameters ()=default
- ∼Parameters ()=default
- Parameters (std::vector< std::string > filenames)

  From a given directory takes all the needed values and creates Parameters object.
- arma::cx_mat getVMatrix (int) const

  V matrix for a given x_i.
- double getE (int) const
- int NX () const
- double getXMin () const
- double getXMax () const
- double getDx () const
- double x (int i) const
- FRIEND_TEST (ParametersOutputTest, x_worksCorrectForNegativeIndices)
- double getUnit () const
- int getNChannels () const
- int getNE () const
- arma::cx_mat getB (int i) const
- int getNSymmetries () const
- int getGrid_points_per_lambda () const
- arma::cx_mat Id () const
- bool isOpen (int nChannel, double energy) const

  Check if the channel is open.
- double kappa (int n1, int n2, int i, double E) const
- double kappa (int n1, int n2, double x, double E) const
- arma::cx_mat getV (double x) const

  Linear interpolation of V (works also for V given on non-constant grid if needed)
- double lambda (double x, double E) const

  de Broglie length for a given potential and x
- double requiredDx (double x, double E) const

Friends

- class Parameters_loadV_Test
- class Parameters_isOpen_Test
- class Parameters_kappaInt_Test
- class Parameters_kappaDouble_Test
- class Parameters_getV_Test
- class Parameters_lambda_Test
- class Parameters_requiredDX_Test

3.2.1 Constructor & Destructor Documentation

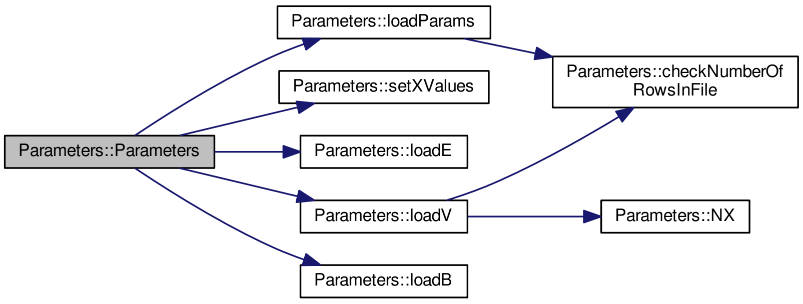
3.2.1.1 Parameters::Parameters ( ) [default]

3.2.1.2 Parameters::∼Parameters ( ) [default]

3.2.1.3 Parameters::Parameters (std::vector< std::string > *filenames*) [explicit]

From a given directory takes all the needed values and creates Parameters object.

Here is the call graph for this function:



3.2.2 Member Function Documentation

3.2.2.1 bool Parameters::checkNumberOfRowsInFile (std::string *filename,* const int *required_number_of_columns*)

3.2.2.2 Parameters::FRIEND_TEST (ParametersInputTest , setXValues_failsIfXMaxLessOrEqualXMin)

3.2.2.3 Parameters::FRIEND_TEST (ParametersInputTest , setXValues_failsIfInvalidDX)

3.2.2.4 Parameters::FRIEND_TEST (ParametersInputTest , setXValues_failsIfInvalidCombinationOfXMinXMaxDx)

3.2.2.5 Parameters::FRIEND_TEST (ParametersInputTest , setXValues_worksGoodForCorrectValues)

3.2.2.6 Parameters::FRIEND_TEST (Parameters_loadV_Test , failsForIncorrectNumberOfRows)

3.2.2.7 Parameters::FRIEND_TEST (Parameters_loadV_Test , worksGoodForGoodFileOneChannel)

3.2.2.8 Parameters::FRIEND_TEST (Parameters_loadV_Test , worksGoodForGoodFileTwoChannels)

3.2.2.9 Parameters::FRIEND_TEST (ParametersInputTest , loadB_failsIfAnyFileDoesNotExistAndPositiveNSymmetries)

3.2.2.10 Parameters::FRIEND_TEST (ParametersInputTest , loadB_worksGoodForGoodFilesOneChannel)

3.2.2.11 Parameters::FRIEND_TEST (ParametersInputTest , loadB_failsIfAnyFileIsIncorrect)

3.2.2.12 Parameters::FRIEND_TEST (ParametersOutputTest , x_worksCorrectForNegativeIndices)

3.2.2.13 arma::cx_mat Parameters::getB (int *i*) const [inline]

3.2.2.14 double Parameters::getDx ( ) const [inline]

3.2.2.15 double Parameters::getE (int *i*) const

3.2.2.16 int Parameters::getGrid_points_per_lambda ( ) const [inline]

3.2.2.17 int Parameters::getNChannels ( ) const [inline]

3.2.2.18 int Parameters::getNE ( ) const [inline]

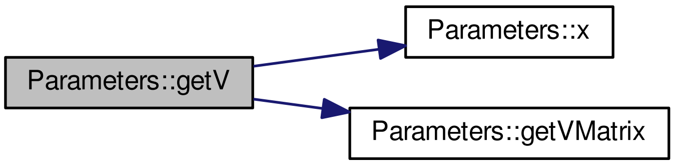
3.2.2.19 int Parameters::getNSymmetries ( ) const [inline]

3.2.2.20 double Parameters::getUnit ( ) const [inline]

3.2.2.21 arma::cx_mat Parameters::getV (double *x*) const

Linear interpolation of V (works also for V given on non-constant grid if needed)

Here is the call graph for this function:



3.2.2.22 arma::cx_mat Parameters::getVMatrix (int *i*) const

V matrix for a given x_i.

3.2.2.23 double Parameters::getXMax ( ) const [inline]

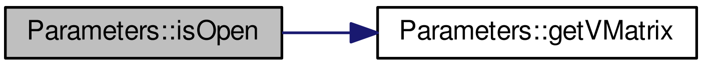
3.2.2.24 double Parameters::getXMin ( ) const [inline]

3.2.2.25 arma::cx_mat Parameters::Id ( ) const [inline]

3.2.2.26 bool Parameters::isOpen (int *nChannel,* double *energy*) const

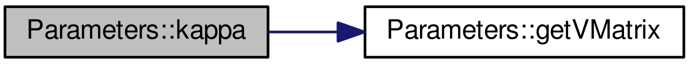
Check if the channel is open.

Here is the call graph for this function:



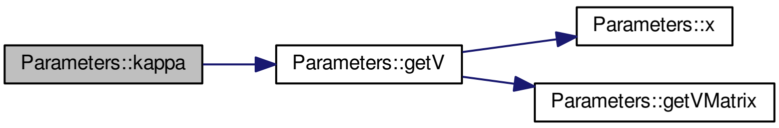
3.2.2.27 double Parameters::kappa (int *n1*, int *n2*, int *i*, double *E*) const

Here is the call graph for this function:



3.2.2.28 double Parameters::kappa (int *n1*, int *n2*, double *x*, double *E*) const

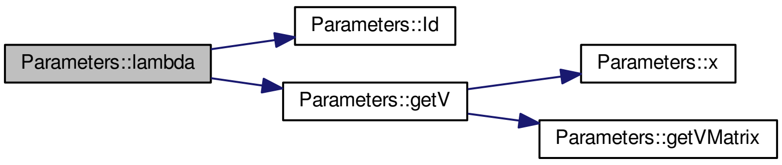
Here is the call graph for this function:



3.2.2.29 double Parameters::lambda (double *x*, double *E*) const

de Broglie length for a given potential and x

Here is the call graph for this function:



3.2.2.30 void Parameters::loadB (std::string *filename* = `"B"`)

Reading the values of B from the file generated in Mathematica.

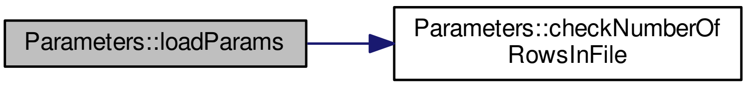
3.2.2.31 void Parameters::loadE (std::string *filename* = `"E.dat"`)

Reading the values of energies from the file generated in Mathematica.

3.2.2.32 void Parameters::loadParams (std::string *filename* = `"Params.txt"`)

Reading the values of parameters from the file generated in Mathematica.

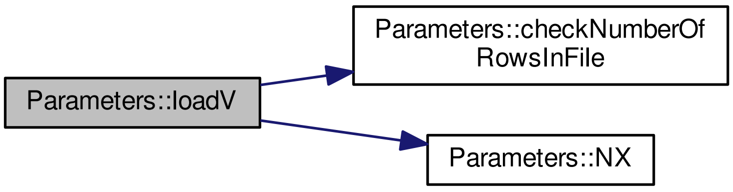
Here is the call graph for this function:



3.2.2.33 void Parameters::loadV (std::string *filename* = `"V.dat"`)

Reading the values of V from the file generated in Mathematica.

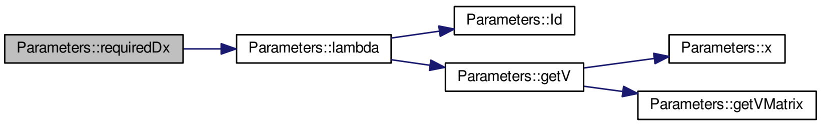
Here is the call graph for this function:



3.2.2.34 int Parameters::NX ( ) const

3.2.2.35 double Parameters::requiredDx (double *x*, double *E*) const

Here is the call graph for this function:



3.2.2.36 void Parameters::setXValues ( )

Setting xValues.

3.2.2.37 double Parameters::x (int *i*) const [inline]

3.2.3 Friends And Related Function Documentation

3.2.3.1 friend class Parameters_getV_Test [friend]

3.2.3.2 friend class Parameters_isOpen_Test [friend]

3.2.3.3 friend class Parameters_kappaDouble_Test [friend]

3.2.3.4 friend class Parameters_kappaInt_Test [friend]

3.2.3.5 friend class Parameters_lambda_Test [friend]

3.2.3.6 friend class Parameters_loadV_Test [friend]

3.2.3.7 friend class Parameters_requiredDX_Test [friend]

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

- Parameters.h
- Parameters.cpp

Chapter 4

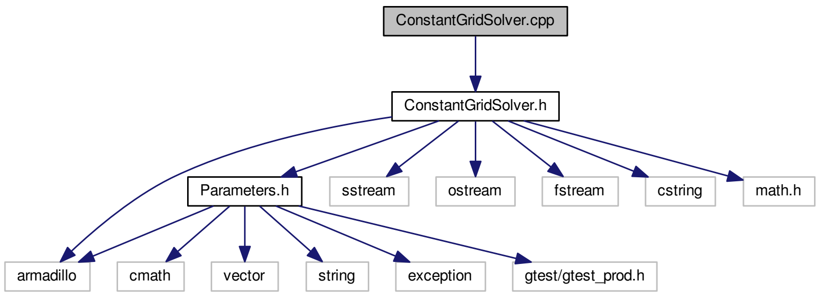
File Documentation

4.1 ConstantGridSolver.cpp File Reference

Definitions of ConstantGridSolver class methods.

```
#include "ConstantGridSolver.h"
```

Include dependency graph for ConstantGridSolver.cpp:



4.1.1 Detailed Description

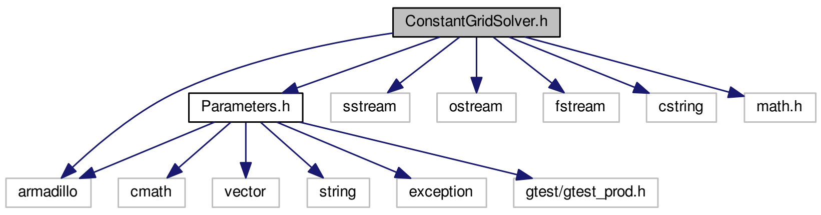
Definitions of ConstantGridSolver class methods.

4.2 ConstantGridSolver.h File Reference

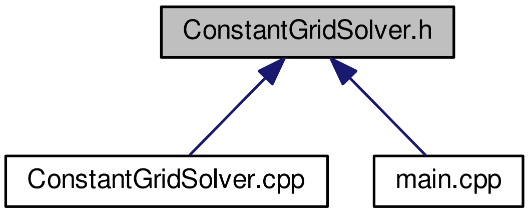
Definition of ConstantGridSolver class.

```
#include "armadillo"
#include "Parameters.h"
#include <sstream>
#include <ostream>
#include <fstream>
#include <cstring>
#include <math.h>
```

Include dependency graph for ConstantGridSolver.h:



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



Classes

- class ConstantGridSolver

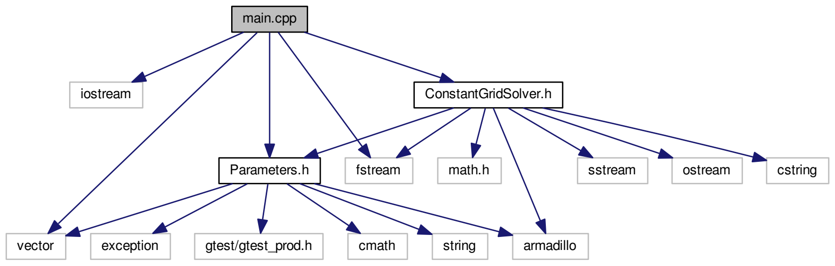
4.2.1 Detailed Description

Definition of ConstantGridSolver class. This file contains a definition of ConstantGridSolver class, performing the calculations for a given set of parameters (Parameters object).

4.3 main.cpp File Reference

```
#include <iostream>
#include <vector>
#include <fstream>
#include "Parameters.h"
#include "ConstantGridSolver.h"
```

Include dependency graph for main.cpp:



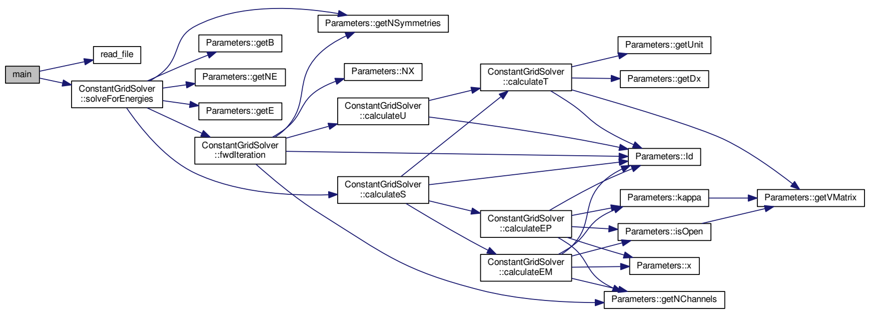
Functions

- std::vector< std::string > read_file (std::string filename)
- int main ()

4.3.1 Function Documentation

4.3.1.1 int main ( )

Here is the call graph for this function:


main
read_file
ConstantGridSolver ::solveForEnergies
Parameters::getB
Parameters::getNE
Parameters::getE
ConstantGridSolver ::fwdIteration
Parameters::getNSymmetries
Parameters::NX
ConstantGridSolver ::calculateU
ConstantGridSolver ::calculateS
ConstantGridSolver ::calculateT
ConstantGridSolver ::calculateEP
ConstantGridSolver ::calculateEM
Parameters::getUnit
Parameters::getDx
Parameters::Id
Parameters::kappa
Parameters::getVMatrix
Parameters::isOpen
Parameters::x
Parameters::getNChannels

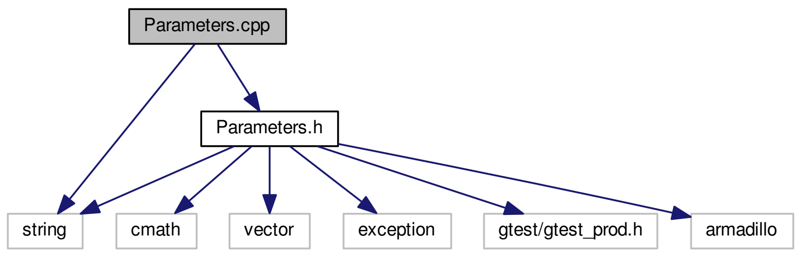
4.3.1.2 std::vector<std::string> read_file (std::string *filename*)

4.4 Parameters.cpp File Reference

Definitions of Parameters class methods.

```
#include <string>
#include "Parameters.h"
```

Include dependency graph for Parameters.cpp:



4.4.1 Detailed Description

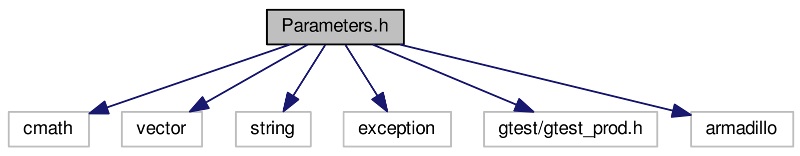
Definitions of Parameters class methods.

4.5 Parameters.h File Reference

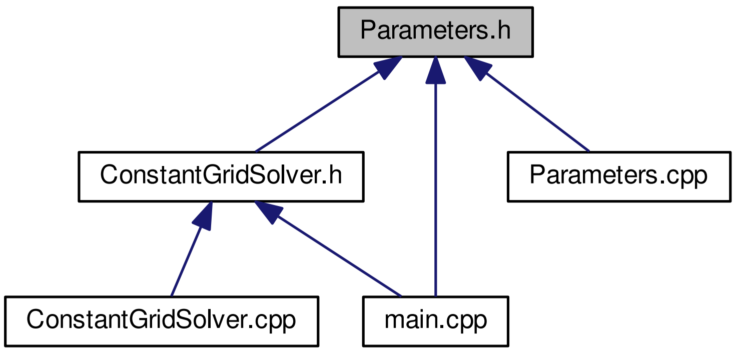
Definition of Parameters class.

```
#include <cmath>
#include <vector>
#include <string>
#include <exception>
#include <gtest/gtest_prod.h>
#include "armadillo"
```

Include dependency graph for Parameters.h:



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



Classes

- class Parameters

4.5.1 Detailed Description

Definition of Parameters class. This file contains the definition of Parameters class.

Index