

# My Project

Generated by Doxygen 1.8.6

Wed Aug 8 2018 12:59:48



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	ConstantGridSolver Class Reference . . . . .	5
3.1.1	Constructor & Destructor Documentation . . . . .	5
3.1.1.1	ConstantGridSolver . . . . .	5
3.1.1.2	~ConstantGridSolver . . . . .	5
3.1.1.3	ConstantGridSolver . . . . .	6
3.1.2	Member Function Documentation . . . . .	6
3.1.2.1	calculateEM . . . . .	6
3.1.2.2	calculateEP . . . . .	6
3.1.2.3	calculateS . . . . .	7
3.1.2.4	calculateT . . . . .	8
3.1.2.5	calculateU . . . . .	9
3.1.2.6	fwdIteration . . . . .	10
3.1.2.7	modifyCCnj . . . . .	11
3.1.2.8	saveS . . . . .	11
3.1.2.9	setParameters . . . . .	12
3.1.2.10	solveForEnergies . . . . .	12
3.2	Parameters Class Reference . . . . .	12
3.2.1	Constructor & Destructor Documentation . . . . .	14
3.2.1.1	Parameters . . . . .	14
3.2.1.2	~Parameters . . . . .	14
3.2.1.3	Parameters . . . . .	14
3.2.2	Member Function Documentation . . . . .	14
3.2.2.1	checkNumberOfRowsInFile . . . . .	14
3.2.2.2	FRIEND_TEST . . . . .	14
3.2.2.3	FRIEND_TEST . . . . .	14

3.2.2.4	FRIEND_TEST	14
3.2.2.5	FRIEND_TEST	14
3.2.2.6	FRIEND_TEST	14
3.2.2.7	FRIEND_TEST	14
3.2.2.8	FRIEND_TEST	14
3.2.2.9	FRIEND_TEST	14
3.2.2.10	FRIEND_TEST	14
3.2.2.11	FRIEND_TEST	14
3.2.2.12	FRIEND_TEST	14
3.2.2.13	getB	14
3.2.2.14	getDx	14
3.2.2.15	getE	15
3.2.2.16	getGrid_points_per_lambda	15
3.2.2.17	getNChannels	15
3.2.2.18	getNE	15
3.2.2.19	getNSymmetries	15
3.2.2.20	getUnit	15
3.2.2.21	getV	15
3.2.2.22	getVMatrix	15
3.2.2.23	getXMax	15
3.2.2.24	getXMin	15
3.2.2.25	Id	15
3.2.2.26	isOpen	15
3.2.2.27	kappa	16
3.2.2.28	kappa	16
3.2.2.29	lambda	16
3.2.2.30	loadB	16
3.2.2.31	loadE	16
3.2.2.32	loadParams	17
3.2.2.33	loadV	17
3.2.2.34	NX	17
3.2.2.35	requiredDx	17
3.2.2.36	setXValues	17
3.2.2.37	x	18
3.2.3	Friends And Related Function Documentation	18
3.2.3.1	Parameters_getV_Test	18
3.2.3.2	Parameters_isOpen_Test	18
3.2.3.3	Parameters_kappaDouble_Test	18
3.2.3.4	Parameters_kappaInt_Test	18
3.2.3.5	Parameters_lambda_Test	18

3.2.3.6	Parameters_loadV_Test . . . . .	18
3.2.3.7	Parameters_requiredDX_Test . . . . .	18
<b>4</b>	<b>File Documentation</b>	<b>19</b>
4.1	ConstantGridSolver.cpp File Reference . . . . .	19
4.1.1	Detailed Description . . . . .	19
4.2	ConstantGridSolver.h File Reference . . . . .	19
4.2.1	Detailed Description . . . . .	20
4.3	main.cpp File Reference . . . . .	20
4.3.1	Function Documentation . . . . .	21
4.3.1.1	main . . . . .	21
4.3.1.2	read_file . . . . .	21
4.4	Parameters.cpp File Reference . . . . .	21
4.4.1	Detailed Description . . . . .	22
4.5	Parameters.h File Reference . . . . .	22
4.5.1	Detailed Description . . . . .	23
<b>Index</b>		<b>24</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">ConstantGridSolver</a> . . . . .	5
<a href="#">Parameters</a> . . . . .	12





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">ConstantGridSolver.cpp</a>	
Definitions of <a href="#">ConstantGridSolver</a> class methods . . . . .	19
<a href="#">ConstantGridSolver.h</a>	
Definition of <a href="#">ConstantGridSolver</a> class . . . . .	19
<a href="#">main.cpp</a> . . . . .	20
<a href="#">Parameters.cpp</a>	
Definitions of <a href="#">Parameters</a> class methods . . . . .	21
<a href="#">Parameters.h</a>	
Definition of <a href="#">Parameters</a> class . . . . .	22



## 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:

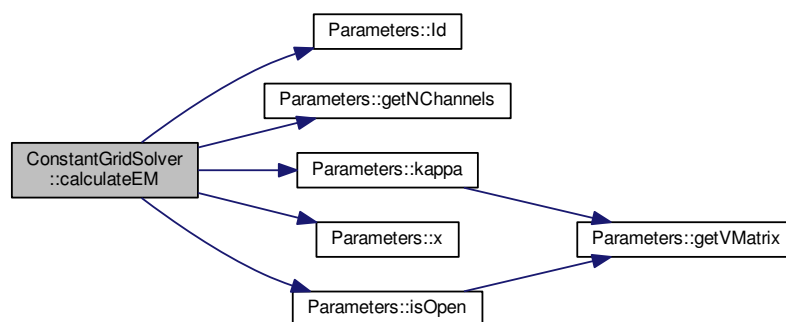
- $\mathbf{E}_{n,n}^-(x_j, E) = \exp(-ikx_j)$  if channel  $n$  is open
- $\mathbf{E}_{n,n}^-(x_j, E) = \cosh(kx_j)$  if channel  $n$  is closed
- $\mathbf{E}_{n,m}^-(x_j, E) = 0$  for  $n \neq m$

#### Parameters

$j$	
$E$	

#### Returns

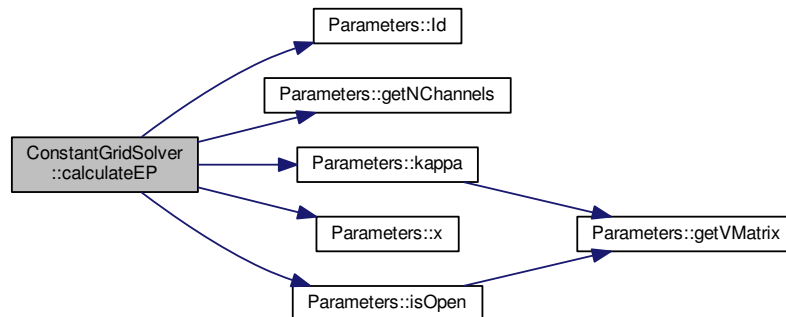
Here is the call graph for this function:



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

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

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

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

Parameters

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

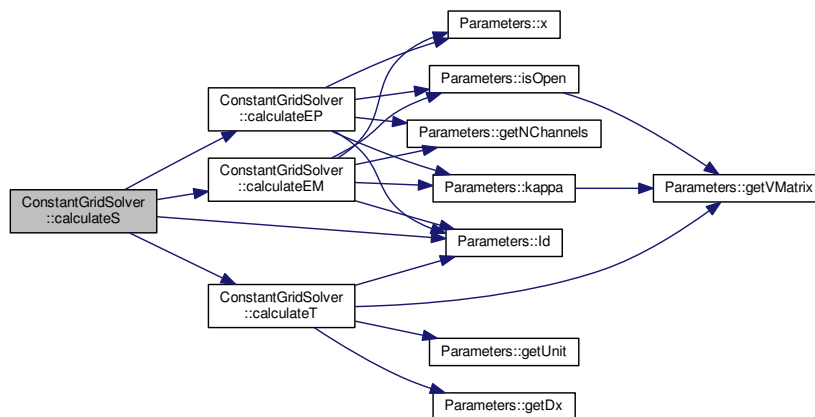
Returns

**S**

Exceptions

<code>std::runtime_error</code>	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:

$$\mathbf{T}_j = -\frac{dx}{12} \mathbf{Q}_j. \quad (3.2)$$

##### Parameters

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

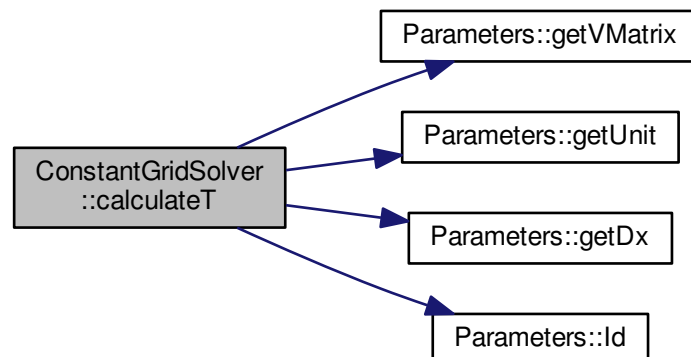
##### Returns

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

##### Exceptions

<code>std::invalid_argument</code>	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:

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

##### Parameters

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

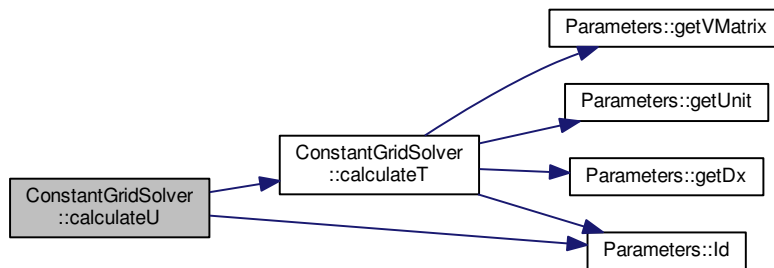
## Returns

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

## Exceptions

<code>std::invalid_argument</code>	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



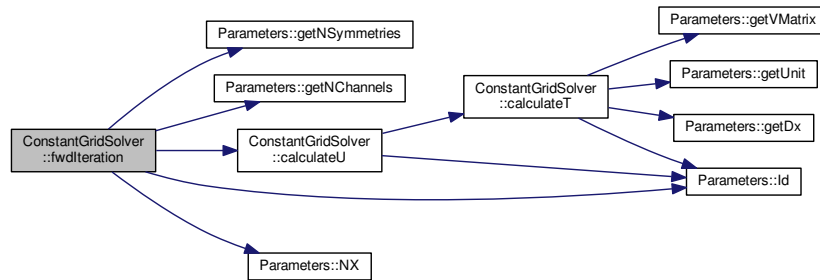
Returns

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

Exceptions

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

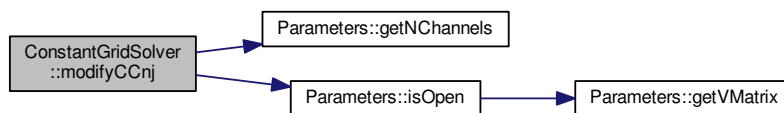
Here is the call graph for this function:



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

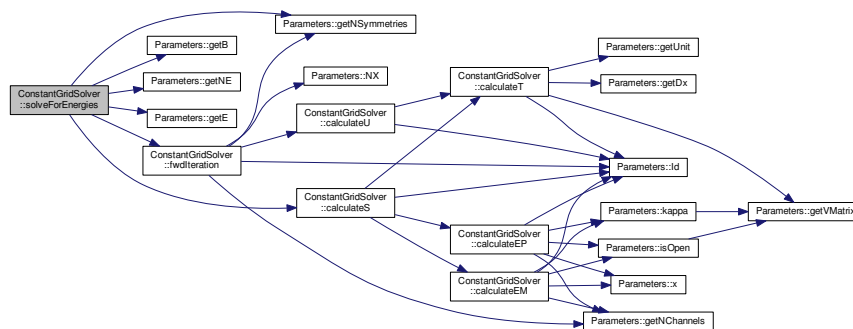
<i>S</i>	- scattering matrix to be saved
<i>E</i>	- energy
<i>directory</i>	- where to save the files

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:



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\_failsIfAnyFilesIsIncorrect)
- 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 [ld](#) () 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

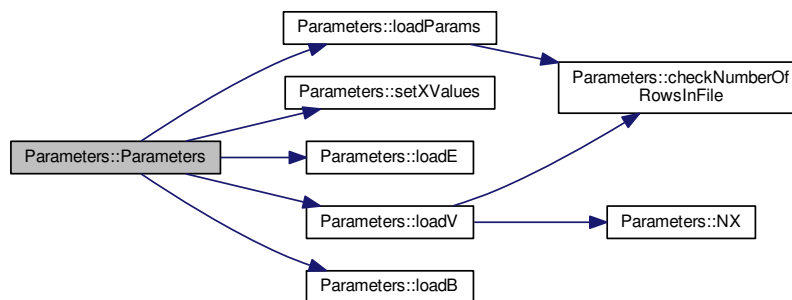
3.2.1.1 `Parameters::Parameters ( )` [default]

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_failsIfAnyFilesIncorrect )`

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]`

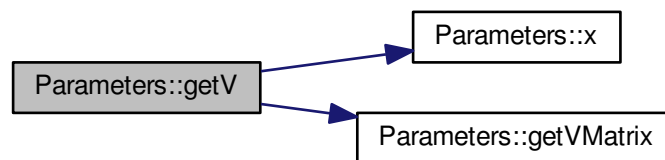
3.2.2.19 `int Parameters::getNSymmetries ( ) const` `[inline]`

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 `xi`.

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::ld ( ) 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`

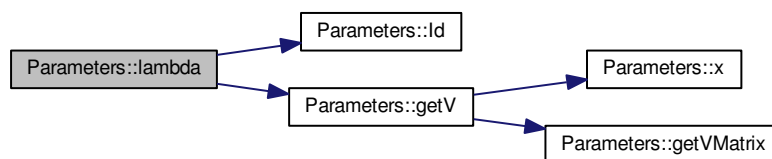
Here is the call graph for this function:



### 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.

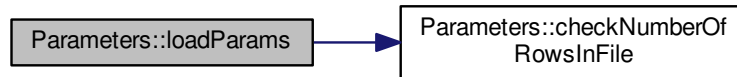
### 3.2.2.31 `void Parameters::loadE ( std::string filename = "E.dat" )`

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.

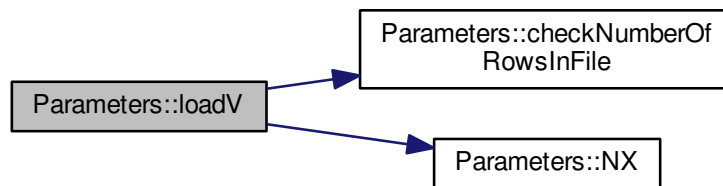
Here is the call graph for this function:



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

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

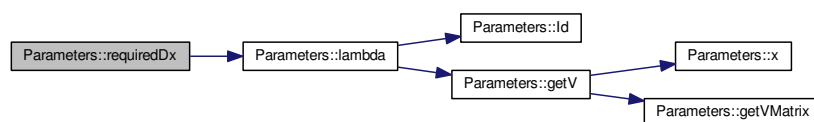
Here is the call graph for this function:



### 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

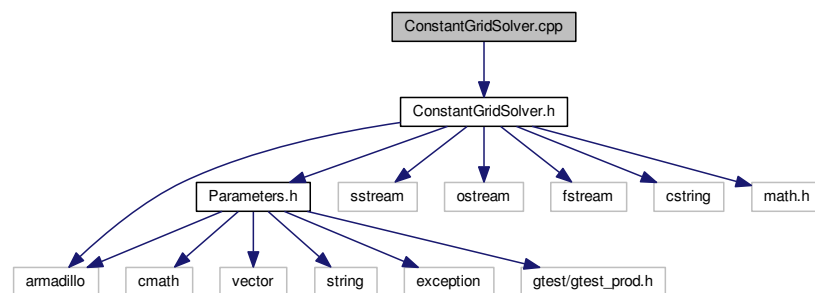
# File Documentation

### 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

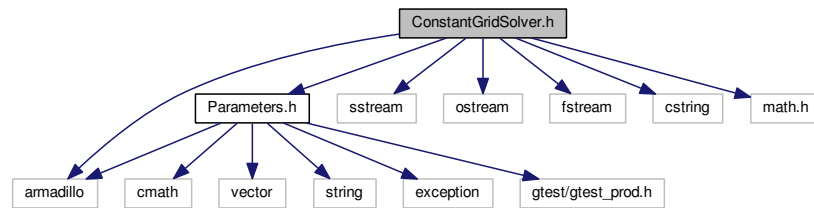
Definitions of [ConstantGridSolver](#) class methods.

### 4.2 ConstantGridSolver.h File Reference

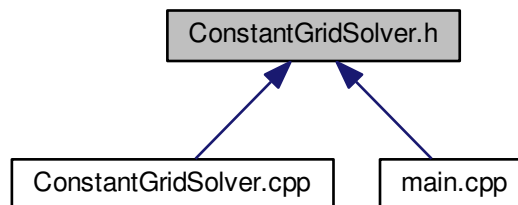
Definition of [ConstantGridSolver](#) class.

```
#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).

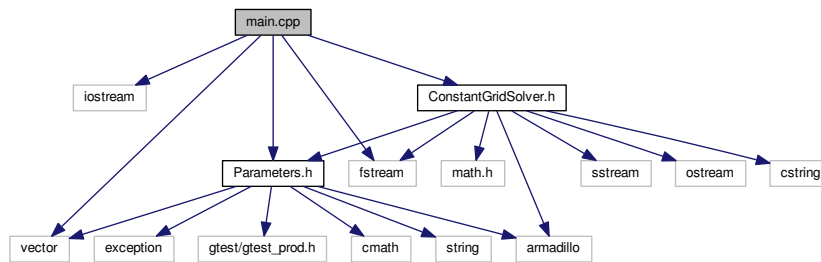
## 4.3 main.cpp File Reference

```

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

```

Include dependency graph for main.cpp:



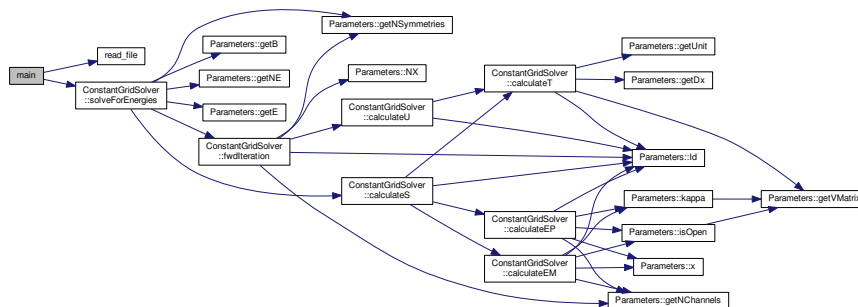
## Functions

- `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:



#### 4.3.1.2 `std::vector<std::string> read_file ( std::string filename )`

## 4.4 Parameters.cpp File Reference

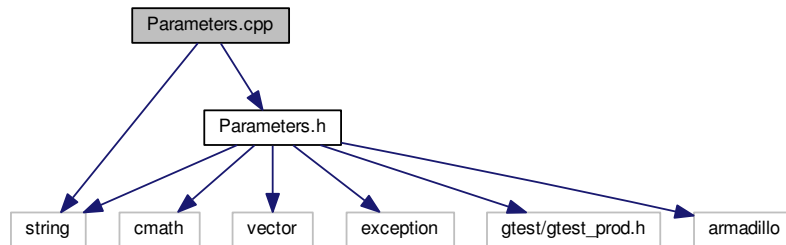
Definitions of `Parameters` class methods.

```

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

```

Include dependency graph for Parameters.cpp:



#### 4.4.1 Detailed Description

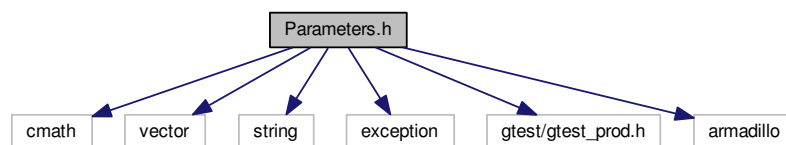
Definitions of [Parameters](#) class methods.

## 4.5 Parameters.h File Reference

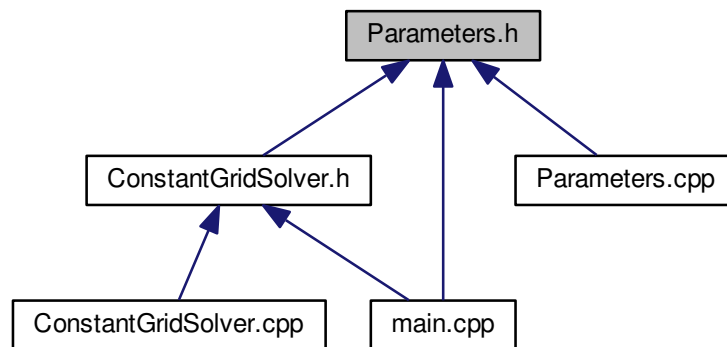
Definition of [Parameters](#) class.

```
#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

- ~ConstantGridSolver
  - ConstantGridSolver, [5](#)
- ~Parameters
  - Parameters, [14](#)
- calculateEM
  - ConstantGridSolver, [6](#)
- calculateEP
  - ConstantGridSolver, [6](#)
- calculateS
  - ConstantGridSolver, [7](#)
- calculateT
  - ConstantGridSolver, [8](#)
- calculateU
  - ConstantGridSolver, [9](#)
- checkNumberOfRowsInFile
  - Parameters, [14](#)
- ConstantGridSolver, [5](#)
  - ~ConstantGridSolver, [5](#)
  - calculateEM, [6](#)
  - calculateEP, [6](#)
  - calculateS, [7](#)
  - calculateT, [8](#)
  - calculateU, [9](#)
  - ConstantGridSolver, [5](#)
  - ConstantGridSolver, [5](#)
  - fwdIteration, [10](#)
  - modifyCCnj, [11](#)
  - saveS, [11](#)
  - setParameters, [12](#)
  - solveForEnergies, [12](#)
- ConstantGridSolver.cpp, [19](#)
- ConstantGridSolver.h, [19](#)
- FRIEND\_TEST
  - Parameters, [14](#)
- fwdIteration
  - ConstantGridSolver, [10](#)
- getB
  - Parameters, [14](#)
- getDx
  - Parameters, [14](#)
- getE
  - Parameters, [14](#)
- getGrid\_points\_per\_lambda
  - Parameters, [15](#)
- getNChannels
  - Parameters, [15](#)
- getNE
  - Parameters, [15](#)
- getNSymmetries
  - Parameters, [15](#)
- getUnit
  - Parameters, [15](#)
- getV
  - Parameters, [15](#)
- getVMatrix
  - Parameters, [15](#)
- getXMax
  - Parameters, [15](#)
- getXMin
  - Parameters, [15](#)
- Id
  - Parameters, [15](#)
- isOpen
  - Parameters, [15](#)
- kappa
  - Parameters, [16](#)
- lambda
  - Parameters, [16](#)
- loadB
  - Parameters, [16](#)
- loadE
  - Parameters, [16](#)
- loadParams
  - Parameters, [16](#)
- loadV
  - Parameters, [17](#)
- main
  - main.cpp, [21](#)
- main.cpp, [20](#)
  - main, [21](#)
  - read\_file, [21](#)
- modifyCCnj
  - ConstantGridSolver, [11](#)
- NX
  - Parameters, [17](#)
- Parameters, [12](#)
  - ~Parameters, [14](#)
  - checkNumberOfRowsInFile, [14](#)
  - FRIEND\_TEST, [14](#)
  - getB, [14](#)
  - getDx, [14](#)
  - getE, [14](#)

- getGrid\_points\_per\_lambda, [15](#)
- getNChannels, [15](#)
- getNE, [15](#)
- getNSymmetries, [15](#)
- getUnit, [15](#)
- getV, [15](#)
- getVMatrix, [15](#)
- getXMax, [15](#)
- getXMin, [15](#)
- Id, [15](#)
- isOpen, [15](#)
- kappa, [16](#)
- lambda, [16](#)
- loadB, [16](#)
- loadE, [16](#)
- loadParams, [16](#)
- loadV, [17](#)
- NX, [17](#)
- Parameters, [14](#)
- Parameters\_getV\_Test, [18](#)
- Parameters\_isOpen\_Test, [18](#)
- Parameters\_kappaDouble\_Test, [18](#)
- Parameters\_kappaInt\_Test, [18](#)
- Parameters\_lambda\_Test, [18](#)
- Parameters\_loadV\_Test, [18](#)
- Parameters\_requiredDX\_Test, [18](#)
- requiredDx, [17](#)
- setXValues, [17](#)
- x, [17](#)
- Parameters.cpp, [21](#)
- Parameters.h, [22](#)
- Parameters\_getV\_Test
  - Parameters, [18](#)
- Parameters\_isOpen\_Test
  - Parameters, [18](#)
- Parameters\_kappaDouble\_Test
  - Parameters, [18](#)
- Parameters\_kappaInt\_Test
  - Parameters, [18](#)
- Parameters\_lambda\_Test
  - Parameters, [18](#)
- Parameters\_loadV\_Test
  - Parameters, [18](#)
- Parameters\_requiredDX\_Test
  - Parameters, [18](#)
- read\_file
  - main.cpp, [21](#)
- requiredDx
  - Parameters, [17](#)
- saveS
  - ConstantGridSolver, [11](#)
- setParameters
  - ConstantGridSolver, [12](#)
- setXValues
  - Parameters, [17](#)
- solveForEnergies
  - ConstantGridSolver, [12](#)

x

Parameters, [17](#)