

# My Project

Generated by Doxygen 1.8.6

Wed Aug 8 2018 12:20:23



# 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 . . . . .	6
3.1.2.4	calculateT . . . . .	7
3.1.2.5	calculateU . . . . .	8
3.1.2.6	fwdIteration . . . . .	9
3.1.2.7	modifyCCnj . . . . .	10
3.1.2.8	saveS . . . . .	10
3.1.2.9	setParameters . . . . .	10
3.1.2.10	solveForEnergies . . . . .	10
3.2	Parameters Class Reference . . . . .	11
3.2.1	Constructor & Destructor Documentation . . . . .	12
3.2.1.1	Parameters . . . . .	12
3.2.1.2	~Parameters . . . . .	12
3.2.1.3	Parameters . . . . .	12
3.2.2	Member Function Documentation . . . . .	13
3.2.2.1	checkNumberOfRowsInFile . . . . .	13
3.2.2.2	FRIEND_TEST . . . . .	13
3.2.2.3	FRIEND_TEST . . . . .	13

3.2.2.4	FRIEND_TEST	13
3.2.2.5	FRIEND_TEST	13
3.2.2.6	FRIEND_TEST	13
3.2.2.7	FRIEND_TEST	13
3.2.2.8	FRIEND_TEST	13
3.2.2.9	FRIEND_TEST	13
3.2.2.10	FRIEND_TEST	13
3.2.2.11	FRIEND_TEST	13
3.2.2.12	FRIEND_TEST	13
3.2.2.13	getB	13
3.2.2.14	getDx	13
3.2.2.15	getE	13
3.2.2.16	getGrid_points_per_lambda	13
3.2.2.17	getNChannels	13
3.2.2.18	getNE	13
3.2.2.19	getNSymmetries	14
3.2.2.20	getUnit	14
3.2.2.21	getV	14
3.2.2.22	getVMatrix	14
3.2.2.23	getXMax	14
3.2.2.24	getXMin	14
3.2.2.25	Id	14
3.2.2.26	isOpen	14
3.2.2.27	kappa	15
3.2.2.28	kappa	15
3.2.2.29	lambda	15
3.2.2.30	loadB	15
3.2.2.31	loadE	15
3.2.2.32	loadParams	16
3.2.2.33	loadV	16
3.2.2.34	NX	16
3.2.2.35	requiredDx	16
3.2.2.36	setXValues	16
3.2.2.37	x	17
3.2.3	Friends And Related Function Documentation	17
3.2.3.1	Parameters_getV_Test	17
3.2.3.2	Parameters_isOpen_Test	17
3.2.3.3	Parameters_kappaDouble_Test	17
3.2.3.4	Parameters_kappaInt_Test	17
3.2.3.5	Parameters_lambda_Test	17

3.2.3.6	Parameters_loadV_Test . . . . .	17
3.2.3.7	Parameters_requiredDX_Test . . . . .	17
<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> . . . . .	<a href="#">5</a>
<a href="#">Parameters</a> . . . . .	<a href="#">11</a>





## 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 std::string S\_type, 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]

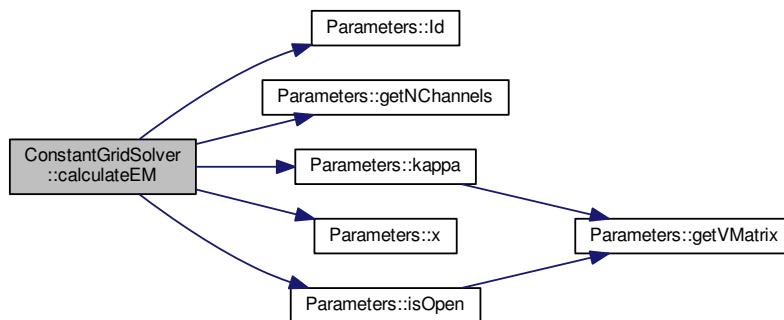
Constructor.

## 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.

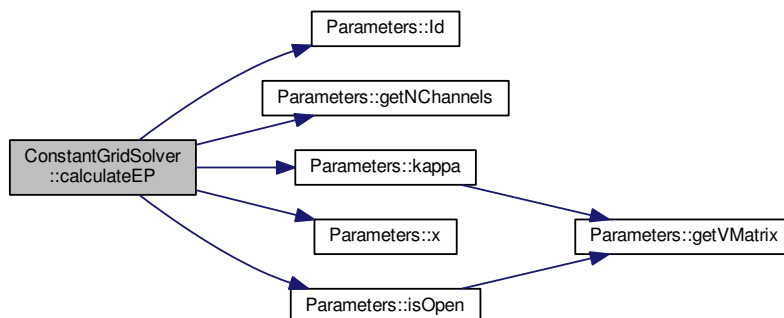
Here is the call graph for this function:



### 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

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

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

Parameters

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

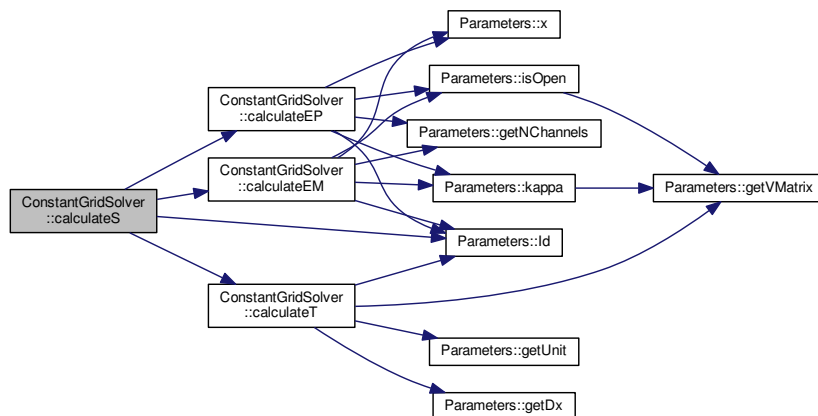
Returns

$\mathbf{S}$

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

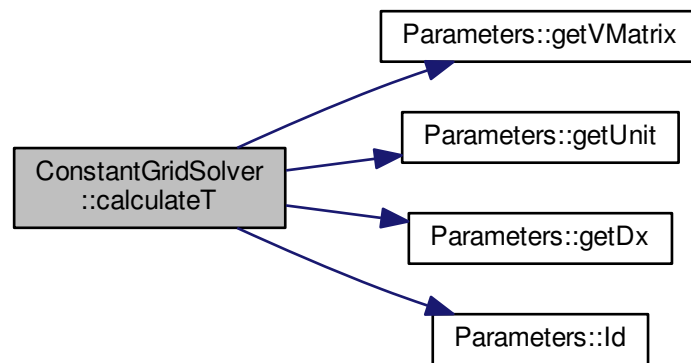
Calculates  $\mathbf{T}(x_j, E)$  matrix.

This method calculates T matrix for given index j and set of parameters [in] j - index of x value for which T is calculated [in] params - [Parameters](#) object for which T is calculated

**Returns**

T matrix

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.

**Parameters**

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

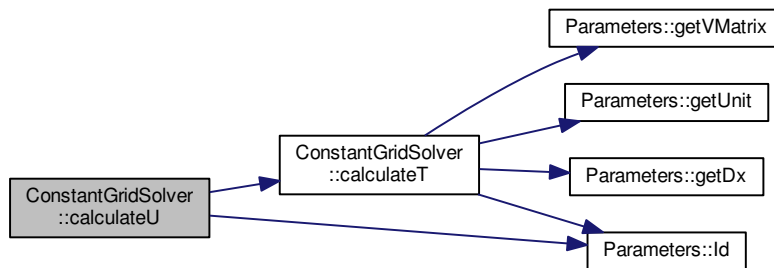
## Returns

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

## 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

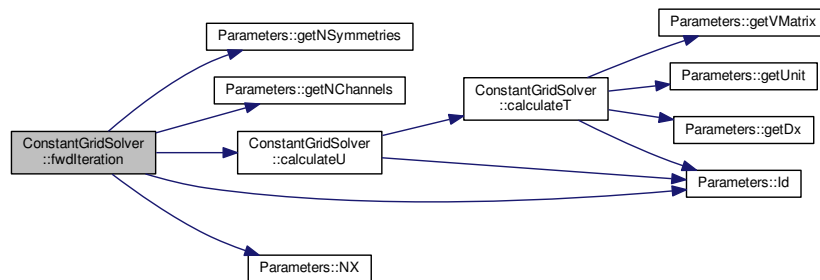
Returns

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

Exceptions

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

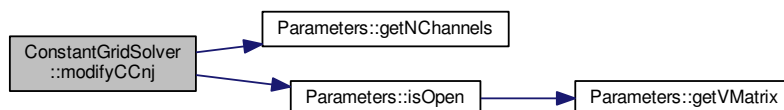
Here is the call graph for this function:



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 std::string *S\_type*, const int *E*, const std::string *directory* )

Saves **S** matrix (Im and Re part separately).

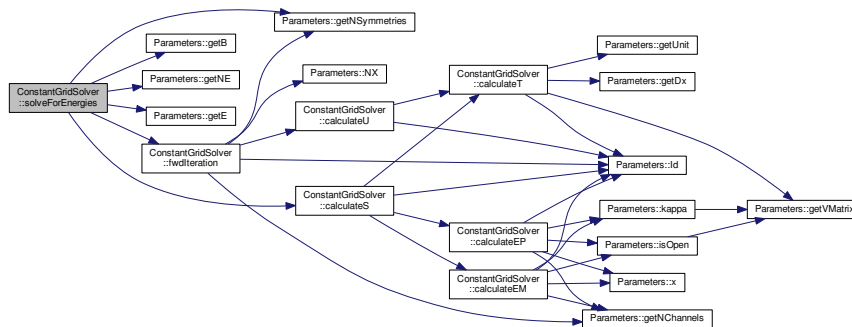
3.1.2.9 void ConstantGridSolver::setParameters ( const Parameters & *parameters* ) [inline]

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\_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 [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

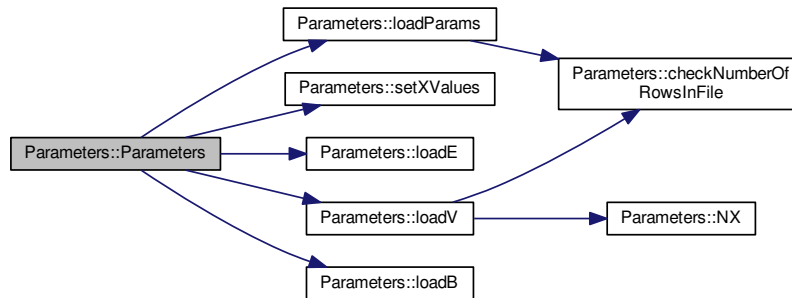
3.2.1.1 [Parameters::Parameters](#) ( ) [default]

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]`

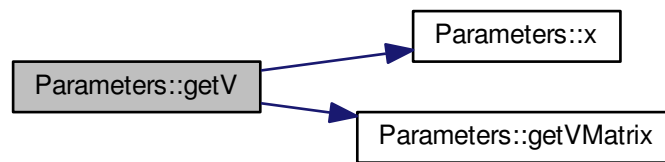
3.2.2.19 `int Parameters::getNSymmetries ( ) const` `[inline]`

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 `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::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`

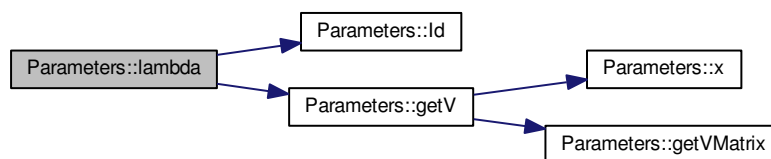
Here is the call graph for this function:



### 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.

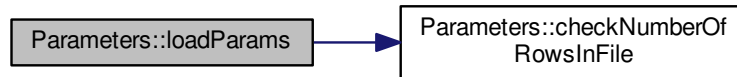
### 3.2.2.31 `void Parameters::loadE ( std::string filename = "E.dat " )`

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.

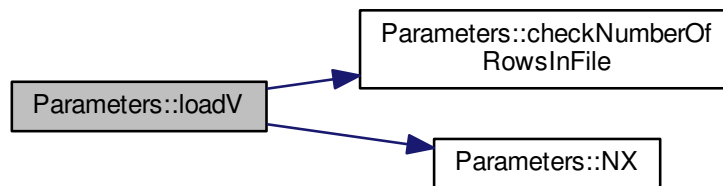
Here is the call graph for this function:



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

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

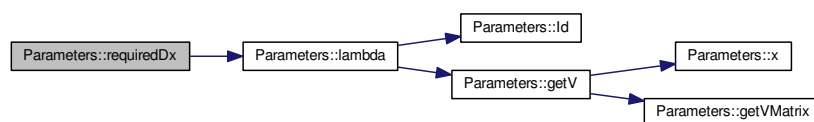
Here is the call graph for this function:



**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

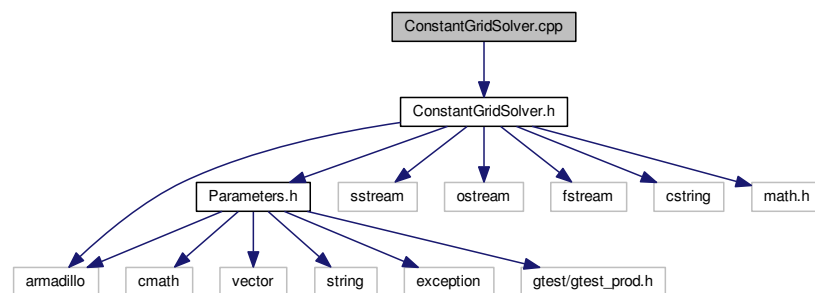
# File Documentation

### 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

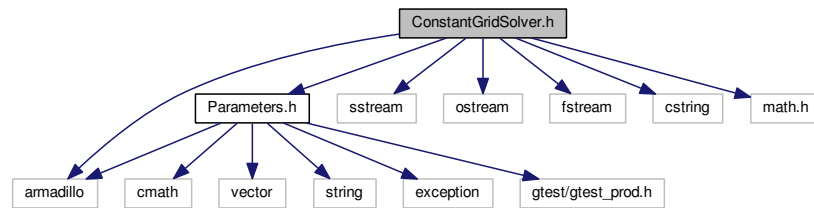
Definitions of [ConstantGridSolver](#) class methods.

### 4.2 ConstantGridSolver.h File Reference

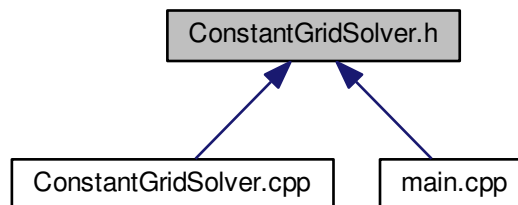
Definition of [ConstantGridSolver](#) class.

```
#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).

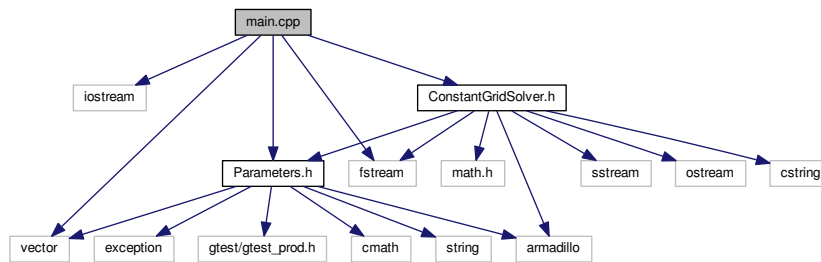
## 4.3 main.cpp File Reference

```

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

```

Include dependency graph for main.cpp:



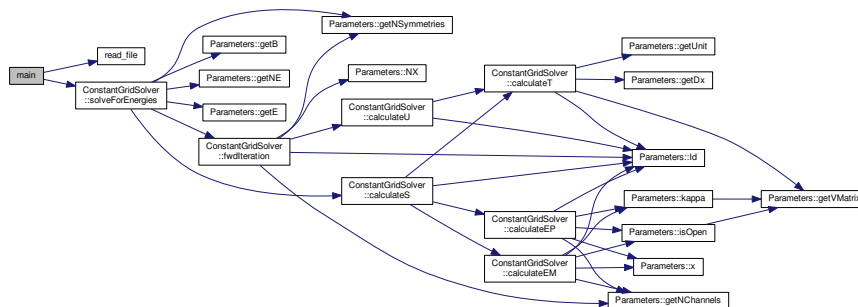
## Functions

- `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:



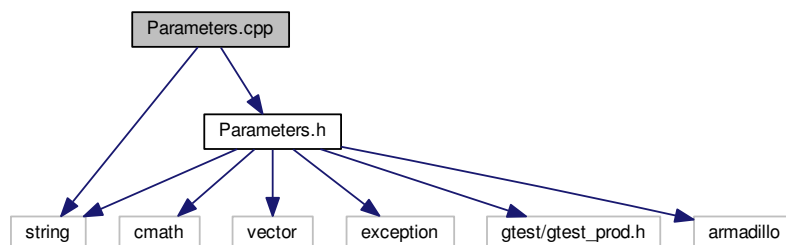
#### 4.3.1.2 std::vector<std::string> read\_file ( std::string filename )

## 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

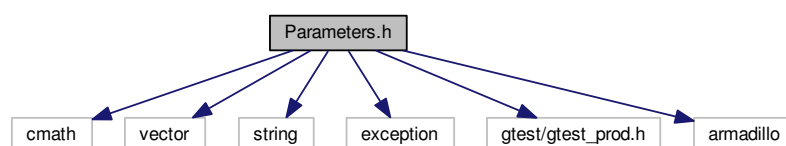
Definition of [Parameters](#) class.

```

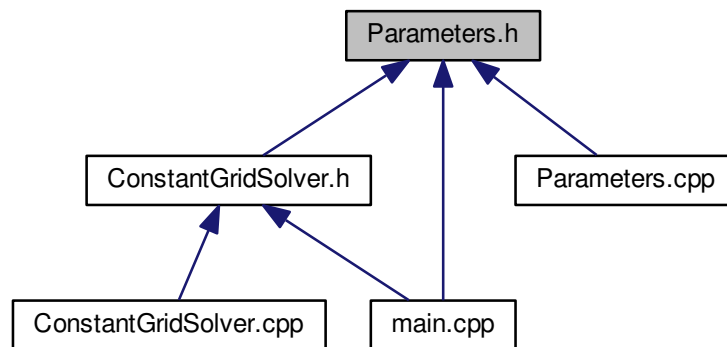
#include <cmath>
#include <vector>
#include <string>
#include <exception>
#include <gtest/gtest_prod.h>
#include "armadillo"

```

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, [12](#)
- calculateEM
  - ConstantGridSolver, [6](#)
- calculateEP
  - ConstantGridSolver, [6](#)
- calculateS
  - ConstantGridSolver, [6](#)
- calculateT
  - ConstantGridSolver, [7](#)
- calculateU
  - ConstantGridSolver, [8](#)
- checkNumberOfRowsInFile
  - Parameters, [13](#)
- ConstantGridSolver, [5](#)
  - ~ConstantGridSolver, [5](#)
  - calculateEM, [6](#)
  - calculateEP, [6](#)
  - calculateS, [6](#)
  - calculateT, [7](#)
  - calculateU, [8](#)
  - ConstantGridSolver, [5](#)
  - ConstantGridSolver, [5](#)
  - fwdIteration, [9](#)
  - modifyCCnj, [10](#)
  - saveS, [10](#)
  - setParameters, [10](#)
  - solveForEnergies, [10](#)
- ConstantGridSolver.cpp, [19](#)
- ConstantGridSolver.h, [19](#)
- FRIEND\_TEST
  - Parameters, [13](#)
- fwdIteration
  - ConstantGridSolver, [9](#)
- getB
  - Parameters, [13](#)
- getDx
  - Parameters, [13](#)
- getE
  - Parameters, [13](#)
- getGrid\_points\_per\_lambda
  - Parameters, [13](#)
- getNChannels
  - Parameters, [13](#)
- getNE
  - Parameters, [13](#)
- getNSymmetries
  - Parameters, [13](#)
- getUnit
  - Parameters, [14](#)
- getV
  - Parameters, [14](#)
- getVMatrix
  - Parameters, [14](#)
- getXMax
  - Parameters, [14](#)
- getXMin
  - Parameters, [14](#)
- Id
  - Parameters, [14](#)
- isOpen
  - Parameters, [14](#)
- kappa
  - Parameters, [14](#), [15](#)
- lambda
  - Parameters, [15](#)
- loadB
  - Parameters, [15](#)
- loadE
  - Parameters, [15](#)
- loadParams
  - Parameters, [15](#)
- loadV
  - Parameters, [16](#)
- main
  - main.cpp, [21](#)
- main.cpp, [20](#)
  - main, [21](#)
  - read\_file, [21](#)
- modifyCCnj
  - ConstantGridSolver, [10](#)
- NX
  - Parameters, [16](#)
- Parameters, [11](#)
  - ~Parameters, [12](#)
  - checkNumberOfRowsInFile, [13](#)
  - FRIEND\_TEST, [13](#)
  - getB, [13](#)
  - getDx, [13](#)
  - getE, [13](#)

- getGrid\_points\_per\_lambda, [13](#)
- getNChannels, [13](#)
- getNE, [13](#)
- getNSymmetries, [13](#)
- getUnit, [14](#)
- getV, [14](#)
- getVMatrix, [14](#)
- getXMax, [14](#)
- getXMin, [14](#)
- Id, [14](#)
- isOpen, [14](#)
- kappa, [14](#), [15](#)
- lambda, [15](#)
- loadB, [15](#)
- loadE, [15](#)
- loadParams, [15](#)
- loadV, [16](#)
- NX, [16](#)
- Parameters, [12](#)
- Parameters\_getV\_Test, [17](#)
- Parameters\_isOpen\_Test, [17](#)
- Parameters\_kappaDouble\_Test, [17](#)
- Parameters\_kappaInt\_Test, [17](#)
- Parameters\_lambda\_Test, [17](#)
- Parameters\_loadV\_Test, [17](#)
- Parameters\_requiredDX\_Test, [17](#)
- requiredDx, [16](#)
- setXValues, [16](#)
- x, [16](#)
- Parameters.cpp, [21](#)
- Parameters.h, [22](#)
- Parameters\_getV\_Test
  - Parameters, [17](#)
- Parameters\_isOpen\_Test
  - Parameters, [17](#)
- Parameters\_kappaDouble\_Test
  - Parameters, [17](#)
- Parameters\_kappaInt\_Test
  - Parameters, [17](#)
- Parameters\_lambda\_Test
  - Parameters, [17](#)
- Parameters\_loadV\_Test
  - Parameters, [17](#)
- Parameters\_requiredDX\_Test
  - Parameters, [17](#)
- read\_file
  - main.cpp, [21](#)
- requiredDx
  - Parameters, [16](#)
- saveS
  - ConstantGridSolver, [10](#)
- setParameters
  - ConstantGridSolver, [10](#)
- setXValues
  - Parameters, [16](#)
- solveForEnergies
  - ConstantGridSolver, [10](#)

x

Parameters, [16](#)